Architecture for Multi-Technology Real-Time Location Systems

Javier Rodas González

Doctoral Thesis UDC / 2015

Director: Carlos José Escudero Cascón

Doutorado en Tecnoloxías da Información e Comunicacións
en Redes Móbiles

UNIVERSIDADE DA CORUÑA

D. Carlos José Escudero Cascón

CERTIFICA

Que la memoria titulada “Architecture for Multi-Technology Real-Time Location Systems” fue realizada por D. Javier Rodas González bajo mi dirección en el Departamento de Electrónica y Sistemas de la Universidade da Coruña, y termina la Tesis que presenta para optar al grado de Doctor.

A Coruña, Septiembre de 2015.

Firmado: D. Carlos José Escudero Cascón
Director de la Tesis Doctoral
Profesor de Universidad
Departamento de Electrónica y Sistemas
Universidade da Coruña

Tesis Doctoral: Architecture for Multi-Technology Real-Time Location Systems

Autor: D. Javier Rodas González

Director: D. Carlos José Escudero Cascón

Fecha de la defensa:

Tribunal

Presidente:

Vocal:

Secretario:

Acknowledgements

First of all, I would like to specially thank my parents for their unconditionally love and support in every single stage of my life. They gave me the opportunity to have one of the best academic preparations that someone could imagine.

I would also like to express my gratitude to my PhD supervisor, Carlos Escudero for his great advices, patience and support, which gave me the opportunity to work at the University of A Coruña for eight years doing what I most love, and pushing my skills to a much higher level.

To Joaquín Míguez and Katrín Achutegui for their contribution to my thesis, with extraordinarily high quality results and publications. Without their contribution this thesis would never be possible.

To Julio Bregáins for his lectures and for giving me the opportunity to discover a so important field of the telecommunications which is the design of antennas. Without his help part of this thesis would also not be possible.

I would also like to thank to Luis Castedo for his fantastic lectures and advices, and not less important, for his financial support during all this years at the University, which allowed me to meet great people and discover places in other countries and at other universities, such as Madrid and Vienna, which I will never forget.

I would like to specially thank Markus Rupp and Sebastian Caban for opening me the doors of Vienna, making me feel always like at home when I was there, and for the most unforgettable winter moments someone could imagine at a University. And also thank Robert Langwieser for his lectures and support during my stays in Vienna, and for giving me the opportunity to improve my skills in low-level RF design and measurement.

I would also like to specially thank Jose Antonio for being always a great friend, for the lunches, dinners and relaxing times that made my life easier and made work hours much more agreeable.

Finally, and not less important, I would like to thank Noelia for her encouragement during my studies, and for always being there during the hardest moments.

Abstract

Indoor localization is a problem that has generated much interest in recent years. Proximity marketing, eHealth, smart-parking and smart-cities, security and emergency units, logistics management, or industrial control systems are some pf the sectors that have demanded new Location Based Services (LBSs). These services are usually implemented using Wireless Sensor Networks (WSNs), capable of transmitting and receiving Radio Frequency (RF) signals in order to locate mobile devices attached to vehicles, people, or animals.

While systems based on satellite systems such as GPS work correctly in outdoor scenarios, indoor localization is still a challenging field of study. On one hand, signal propagation problems are common, not only due to reflections and scattering due to the building structures, but also because of signal attenuation and fading caused mainly by people in movement. To overcome these issues, most of the approaches use several WSNs with a combination of multiple wireless technologies, such as WiFi, ZigBee or Bluetooth, some of them also available in mobile devices such as smartphones and tablets. On the other hand, data received from multiple devices must be filtered and combined by means of location algorithms and techniques in order to obtain precise and robust Real-Time Location Systems (RTLSs).

Therefore, it is common to implement hybrid location systems with support for several technologies at the same time. Nevertheless, the development of such systems entails a huge complexity. Thus, one of most widely accepted alternatives is the implementation of software architectures for localization, which provide several benefits. First, accessing to different kinds of hardware devices entails fewer platform and technology restrictions. Second, some common tasks are easier to perform, such as sensor data gathering and storage. Finally, architectures provide utilities for adding and retrieving localization data, user management, or the possibility of using several mapping and coordinate systems.

In this work, we present several solutions for implementing software architectures for localization. First, we propose a mono-technology architecture using only Received Signal Strength (RSS) signal levels for ranging, which evolves into a much more complete multi-technology architecture in a second stage. The proposed approaches implement several functionalities that resolve most of the hybrid RTLS system requirements, such as:

- Multi-technology.
- Support for several coordinate systems and mapping applications.

- Data fusion.
- Protection and security for both data and user access.
- Standardized API for remote access.
- Support for off-line data queries, not only on-line data and in real-time.
- Depending on different user roles, it eases their tasks at different access levels: registration of WSNs, building blueprints, anchor and mobile node networks registration, generic sensor support, addition and retrieval of measurements and raw sensor data, multiple query support for filtered position estimations, etc.

Moreover, we also contributed with different WSN physical layer implementations and experiments. And, due to collaborations with other research groups at different universities we have contributed with a customized hardware and software solution for localization based on RFID technology, as well as with the design of new antenna models based on linear-arrays of Electromagnetically Coupled Patchs (ECPs), valid for improving the WSN communication performance.

Resumo

O problema da localización no interior de edificios foi adquirindo cada vez máis importancia nos últimos anos debido á enorme demanda de novos servizos baseados en localización (LBSs), que apareceron en todo tipo de sectores como eHealth, marketing por proximidade, smart-parking e smart-cities, seguridade e emerxencias, loxística ou control industrial, entre outros. Estes sistemas habitualmente estan baseados na implementación de redes de sensores sen fíos (WSN) capaces de transmitir ou recibir sinais de radio (RF) para localizar dispositivos móbiles, xeralmente adheridos a vehículos, persoas ou animais.

Mentres que en exteriores os sistemas de satélites baseados en tecnoloxías como GPS funcionan correctamente na maioría de entornos, a localización en interiores non é unha tarefa sinxela de resolver e aínda inclúe múltiples retos. Principalmente aparecen problemas de propagación debido ás reflexións e rebotes dos sinais nas estruturas dos edificios, pero tamén debido a atenuacións e apantallamentos ocasionados xeralmente por xente en movemento. Para resolver estes problemas é necesario implementar as redes de sensores utilizando unha ou varias tecnoloxías sen fíos (como WiFi, ZigBee ou Bluetooth), algunhas delas dispoñibles en terminais sen fíos como smartphones ou tablets. Pero, por outra parte, tamén é necesario o uso de múltiples algoritmos e técnicas de localización para filtrar e posiblemente combinar os datos destas tecnoloxías, permitindo obter así sistemas de localización en tempo real (RTLS) robustos e coa maior precisión posible.

Deste xeito, a aproximación máis usual na actualidade para resolver estos problemas é a implementación de sistemas de localización híbridos que soporten múltiples tecnoloxías simultaneamente. Nembargantes, o desenvolvemento destes sistemas leva implícito unha gran complexidade. Unha das alternativas comunmente aceptada é a implementación dunha arquitectura de software para localización, a cal ofrece varias vantaxes. En primeiro lugar, permite minimizar o número de restriccións multi-plataforma e multi-tecnoloxía á hora de acceder a distintos tipos de dispositivos hardware. En segundo lugar, facilítase a realización de tarefas comúns como a recolección e o almacenamento das medicións de sensores. Ademais, proporciónanse mecanismos para inserir e recuperar datos de localización así como xestión de usuarios ou manipulación de múltiples sistemas de mapas e coordenadas.

Neste traballo presentamos varias solucións á hora de implementar arquitecturas de software para localización, comenzando por unha mono-tecnoloxía baseada unicamente na recolección de niveis de sinal RSS, que evoluciona posteriormente a unha arquitectura multi-tecnoloxía.

As solucións propostas ofrecen diferentes funcionalidades que resolven moitos dos problemas asociados aos sistemas híbridos RTLS, entre as que podemos destacar:

- Multi-tecnoloxía.
- Soporte de múltiples sistemas de coordenadas e de aplicacións de mapas.
- Fusión de datos.
- Protección e seguridad, tanto de datos como de acceso de usuarios.
- API estandarizado para acceso remoto.
- Soporte de consultas de datos off-line, non só on-line e en tempo real.
- Facilidade de uso para os diferentes usuarios que utilicen a plataforma mediante chamadas a varios niveis: rexistro de WSNs, planos de edificios, rexistro de redes de áncoras e de nodos móviles, soporte de sensores xenéricos, inserción e consulta de medicións e de datos sensoriais en cru, inserción e consulta de posicións estimadas por algoritmos de localización, etc.

Tamén contribuimos con múltiples implementacións da capa física de WSNs e experimentos. E grazas á colaboración con outros grupos de investigación de diferentes universidades puidemos, por unha parte, contribuír cunha solución de hardware e software para localización baseada en tecnoloxía RFID e, por outra parte, no deseño de novos modelos de antenas baseados en arrays lineais de ECPs, válidos para mellorar o rendemento das comunicacións en WSNs.

Resumen

El problema de la localización en el interior de edificios ha ido adquiriendo cada vez más importancia en los últimos años debido a la enorme demanda de nuevos servicios basados en localización (LBSs), que han ido apareciendo en la industria en sectores de todo tipo como eHealth, marketing por proximidad, smart-parking y smart-cities, seguridad y emergencias, logística o control industrial, entre otros. Estos sistemas habitualmente se basan en la implementación de redes de sensores inalámbricos (WSN) capaces de transmitir o recibir señales de radio (RF) para localizar dispositivos móviles, generalmente adheridos a vehículos, personas o animales.

Mientras que en exteriores los sistemas satelitales basados en tecnologías como GPS funcionan correctamente en la mayoría de entornos, la localización en interiores todavía plantea múltiples retos y no es una tarea sencilla de resolver. Principalmente aparecen problemas de propagación debido a los reflejos y rebotes de las señales en las estructuras de los edificios, pero también debido a atenuaciones y apantallamientos ocasionados generalmente por gente en movimiento. Para resolver estos problemas es necesario implementar las redes de sensores utilizando una o varias tecnologías inalámbricas (como pueden ser WiFi, ZigBee o Bluetooth), algunas de ellas disponibles en terminales inalámbricos como smartphones o tablets. Pero, por otra parte, también es necesario el uso de múltiples algoritmos y técnicas de localización, para filtrar y posiblemente combinar los datos de estas tecnologías, permitiendo obtener así sistemas de localización en tiempo real (RTLS) robustos y con la mayor precisión posible.

De este modo, la aproximación más usual en la actualidad para resolver estos problemas es la implementación de sistemas de localización híbridos que soporten múltiples tecnologías simultáneamente. No obstante, el desarrollo de estos sistemas lleva implícito una gran complejidad. Una de las alternativas comúnmente aceptada es la implementación de una arquitectura de software para localización, que ofrece varias ventajas. En primer lugar, permite minimizar el número de restricciones multi-plataforma y multi-tecnología a la hora de acceder a distintos tipos de dispositivos hardware. En segundo lugar, se facilitan tareas comunes como la recolección y almacenamiento de las mediciones de los sensores. Además, se proveen mecanismos para insertar y recuperar datos de localización así como gestión de usuarios o manejo de múltiples sistemas de mapas y coordenadas.

En este trabajo presentamos varias soluciones a la hora de implementar arquitecturas de software para localización, empezando por una mono-tecnología basada únicamente en la

recoleccion de niveles de señal RSS, que se evoluciona posteriormente a una arquitectura multi-tecnología. Las soluciones propuestas ofrecen diferentes funcionalidades que resuelven muchos de los problemas asociados a los sistemas híbridos RTLS, entre las que podemos destacar:

- Multi-tecnología.
- Soporte de múltiples sistemas de coordenadas y de aplicaciones de mapas.
- Fusión de datos.
- Protección y seguridad, tanto de datos como de acceso de usuarios.
- API estandarizado para acceso remoto.
- Soporte de consultas de datos off-line, no solo on-line y en tiempo real.
- Facilidad de uso para los diferentes usuarios que utilicen la plataforma, mediante llamadas a varios niveles: registro de WSNs, planos de edificios, registro de redes de anchors y de nodos móviles, soporte de sensores genéricos, inserción y consulta de mediciones y de datos sensoriales en crudo, inserción y consulta de posiciones estimadas por algoritmos de localización, etc.

También contribuimos con múltiples implementaciones de la capa física de WSNs y experimentos. Y gracias a la colaboración con otros grupos de investigación de diferentes universidades hemos podido, por una parte, contribuir con una solución de hardware y software para localización basada en tecnología RFID y, por otra parte, en el diseño de nuevos modelos de antenas basados en arrays lineales de ECPs, válidos para mejorar el rendimiento de las comunicaciones en WSNs.

Contents

List of Figures

Chapter 1

Introduction

The problem of RTLSs in indoor scenarios is receiving a great deal of attention because of the many LBSs that can be implemented with them, such as route management [67], guidance [126], points of interest [78, 141], augmented reality [103], healthcare [108] and others. To provide such services we need a positioning system, typically based on radio range communications. The Global Positioning System (GPS) is frequently used outdoors due to its worldwide coverage. Unfortunately, GPS radio waves are unable to penetrate building structures, leaving indoor areas uncovered [111]. Moreover, conventional GPS receivers have a poor accuracy, typically in the order of 10 meters [93, 147]. Thus, other technologies such as WSN are needed to implement Indoor Positioning Systems (IPSs) and enable LBSs in indoor environments.

Back in 2006 when we started researching in the indoor localization field, there were not many hardware alternatives available to implement WSNs, valid for solving indoor tracking problems at a certain reduced cost (which is always desirable in any RTLS system). At that time, when cell phones were not smart and laptops were much less communicative than nowadays, only a few technologies such as Bluetooth Classic and less often WiFi were implemented in such devices, able to interact with such WSNs. At that time, other wireless technologies such as ZigBee and ultra-sounds started to arrive, mainly in the form of development or evaluation kits, which could be used to implement an RTLS in a more non-centralized way.

Nowadays, there is a continuous emergence of these technologies and many kinds of portable devices continuously evolve and can be used for localization purposes. There is an increasing number of mobile phones, smart-phones, Personal Digital Assistants (PDAs), tablets and laptops equipped with even more radio interfaces such as WiFi, Bluetooth Low Energy (BLE), Enhanced GPS (E-GPS) and Assisted GPS (A-GPS), Near Field Communication (NFC), and even inertial sensors, such as an accelerometer and a digital compass. Luckily, we can take advantage of these features for indoor positioning, and use them to directly estimate the position of persons, animals, vehicles, etc.

Many other open-source rapid prototyping solutions that can be combined with RF transceivers also appeared and continued to grow, such as Arduino [43], Raspberry Pi [53], Teensy [55], panSTamp [52], BeagleBone [44] and many others.

While outdoor positioning systems typically operate without many problems using satellite systems, indoor positioning is a non trivial task, as signals which propagate through the indoor environment are scattered, reflected and affected by multi-path fading. Moreover, signals are affected by transient effects such as human bodies absorbing signals [69] or the humidity level. For instance, narrowband technologies working on the 2.4 GHz ISM band, such as WiFi, ZigBee and Bluetooth, are affected similarly by environmental phenomena. They use RSS noisy information for ranging and they have similar accuracy. According to the RADAR project [70] the median accuracy of fingerprint technique in a noisy WiFi channel can be 2.94 m. Subsequent research efforts have tried to improve positioning accuracy by means of several kinds of location and classification algorithms, such as Support Vector Machines [149], Neural Networks [74, 114] and Particle Filters [90, 92, 100]. Motion models have also been used to better model signal fluctuations in non static scenarios [71, 90, 92, 100, 104]. However, the accuracy that can be obtained by using an indoor positioning system cannot only be attributed exclusively to the chosen location algorithm; it is also affected by factors such as the number of sensors, their placement, their sensitivity, the orientation of their antennas or the number of samples used, etc. [104, 112]. Elnahrawy et al. [95] made an extensive study comparing a wide range of location algorithms, and concluded that 10 feet (about 3 m) supposes a feasible lower bound for these indoor systems based on RSS measurements, due to the inherent limitations of distinguishing RSS values at short distances.

Thus, for an LBS demanding greater accuracy, e.g. below one meter, other non-RSS based technologies are needed. Ultrasound and Ultra Wide Band (UWB) technologies appear as feasible options as they use Time-of-Arrival (ToA) and Time-Difference-of-Arrival (TDoA) information, much more precise than RSS, and more robust to multi-path fading and to external interferences. However, the availability of these technologies and their price compared, for instance, to WiFi and Bluetooth, sometimes force us to deploy multiple WSNs to cover a large scenario. UWB platforms require the deployment of a much more expensive infrastructure, especially designed for precise time synchronization between the readers of the network but, on the other hand, it typically uses inexpensive UWB transmitting tags. Another weak point of UWB is that there are almost no portable devices on the market (i.e. a smartphone) which integrate any UWB part. Luckily, the size and weight of the UWB tags are extremely small so they can be easily attached to any other portable device, being able to obtain a much higher accuracy.

To obtain a ubiquitous and reliable RTLS supporting indoor and outdoor scenarios, we usually require hybrid location systems supporting multiple technologies simultaneously, but these are quite complex to implement.

From the hardware point of view all these alternatives have something in common: they severely restrict the RTLS design due to the particularities of each of them when obtaining measurements, as they use different access interfaces. Furthermore, several constraints arise due to software drivers and libraries, which are usually available in one specific operating system or hardware platform that resides in the host that collects and processes the measurement data. Open operating systems usually allow a deeper and freer low-level access (so they are preferable to implement these systems), but in some situations the hardware is only supported by a closed platform.

For these reasons the first RTLS platforms started to be implemented using a single programming language and operating system (the one which gives the maximum flexibility) and we were typically forced to port (rewrite or adapt) the localization algorithm (sometimes quite complex) to that particularly platform, which had already been programmed and tested using high-level simulation environments such as Matlab. This process usually had to be repeated when implementing a new RTLS with different WSN technologies, which is an expensive and time consuming task.

Furthermore, when we needed to get data from the WSN in real time, we were forced to adapt the location algorithm in order to work with stream-type data, a task which usually requires an enormous effort for some programmers or researchers. These people are usually accustomed to working with off-line data, log files or simulated data that is loaded into memory in a single step.

Thus, to solve some of these issues, a good choice is to implement a software architecture that minimizes the number of restrictions and limitations we usually suffer when implementing an RTLS system. This mainly eases the way different level users access the data and interact with other people in charge with different specialized tasks, such as deploying the WSN hardware and providing empirical measurements, etc.

1.1 Main Objectives

In this thesis we focus on designing and implementing two distributed multi-layer software architectures suitable for real-time localization, starting with a limited mono-technology architecture followed by a much more advanced multi-technology architecture. We not only focus on the hardware architecture but also on software design and RTLS experiments. We carried out several measurement campaigns in indoor and outdoor scenarios in which we experimentally evaluated WSNs systems using Bluetooth, ZigBee and GPS technologies. Other simulation results make use of the WiFi, ZigBee, RFID, UWB and inertial sensor technologies.

Thanks to the collaboration with the department of Signal Theory & Communications, Universidad Carlos III de Madrid (UC3M) we were able to evaluate our software architecture

developments and measurement campaigns, in several scenarios and with multiple advanced localization techniques [63–66].

We also collaborated with the Institute of Communications and Radio-Frequency Engineering, Vienna University of Technology, Austria, where we were able to work on building and deploying a Long Term Evolution (LTE) testbed [81–84, 115, 134], and where we also gained experience in low-level RF hardware design. As a result of that collaboration, we implemented a custom hardware solution to reduce the hardware cost of RTLS systems based on Radio-Frequency Identification (RFID) technology. With the proposed solution we minimize the number of RFID readers needed (which are a costly hardware) while maximizing the number of antennas that can be connected to each reader (up to 16 antennas in our proposed solution).

And finally, we also started a collaboration with Julio Brégains, from the Grupo de Tecnología Electrónica y Comunicaciones (GTEC) of the University of A Coruña, who has a proven track record in the antenna design research field. We simulated and implemented new linear array numeric models of ECP antennas, and provided design methodologies that allow these antenna numeric models to be adapted to different frequency ranges, thereby enabling their use to improve the performance of all kind of WSNs technologies used in indoor localization. Therefore, such antenna design cannot only be used to improve the performance of Wireless Local Area Network (WLAN) communications but also that of localization systems based on WSNs using RF transceivers, i.e. those that do not use infra-red, optical or acoustic sensors. We also applied a new prototyping technique during the building of two antenna prototypes, which allowed us to more accurately construct the antennas based on the simulated numerical models, avoiding human imprecisions during their construction.

1.2 Related Publications and Projects

Most of the content of this thesis has been published in different national and international conferences and a Journal Citation Report (JCR) Q1 journal:

- The first single-technology software architecture introduced in Chapter 2 is based on the publication [128], and the experimental implementations and measuring campaigns in this chapter were used in the publications [65, 129, 130].
- The second multi-technology software architecture introduced in Chapter 3 is published in [131].
- The customized hardware solution for localization presented in Chapter 4 is based on the publication [133].
- Finally, Chapter 5, where we present some antenna numeric models, is based on the publications [122, 132] (and a future journal already under review).

Moreover, our ZigBee WSN implementations and additional measurement campaigns also contributed to other international conference publications [63, 64] and a JCR Q1 journal [66].

The research work obtained during the period of this thesis has contributed to the following research projects and was also possible due to the following grants:

- *m:Ciudad*: Funded between 2006 and 2008 by the Spanish Ministerio de Industria, Turismo y Comercio (subprogram PROFIT, with reference: FIT-330503-2006-2).
- *LOCUS*: Funded between 2007 and 2010 by the Galician Consellería de Innovación e Industria, Xunta de Galicia (reference: 07TIC019105PR).
- *PIRAmIDE*: Funded between 2008 and 2010 by the Spanish Ministerio de Industria, Turismo y Comercio (subprogram Avanza I+D, with reference: TSI-020301-2008-2).
- *Integrated Action for Scientific and Technological Research between Spain and Austria*: Funded between 2010 and 2011 by the Spanish Ministerio de Ciencia e Innovación (reference: AT2009-0014).
- *MISCOM*: Funded between 2010 and 2012 by the Spanish Ministerio de Ciencia e Innovación (subprogram INNPACTO, with reference: IPT-020000-2010-035).
- *GNUTEST*: Funded between 2010 and 2013 by the Galician Consellería de Economía e Industria, Xunta de Galicia (reference: 10TIC003CT).
- *TECRAIL*: Funded between 2011 and 2014 by the Spanish Ministerio de Ciencia e Innovación (subprogram INNPACTO, with reference: IPT-2011-1034-370000).

1.3 Thesis Overview

This thesis is organized as follows. In Chapter 2 we address the problem of implementing a distributed multi-layer software architecture. This was our first architecture implementation and is limited only to RSS based RTLS systems and to a single technology at a time, but it allowed us to perform many experiments and obtain measurement campaigns. In particular we use different Bluetooth Classic and ZigBee hardware for the physical layer implementations, and show some experimental results.

In Chapter 3 we evolve the previous work on software architecture for implementing RTLS systems to a completely new level. We propose an approach to the problem developing a generic and easy-to-use hybrid RTLS by introducing a distributed multi-layer and multi-technology software architecture. The proposed solution is not only able to obtain measurements from several WSNs and several generic sensors (i.e. inertial sensors, motion detector, video cameras, etc.), but also to obtain position estimations by means of several location algorithms running

simultaneously in real-time. It also provides a flexible data fusion module to provide combined positioning information to several concurrent client applications. This fusion module can automatically merge positioning information from an arbitrary number of nodes (several WSN technologies) and sensors registered in the system. Moreover, communication between the clients and the server is made by using a well defined API interface, which provides mapping information with different coordinate system support, user access control and data persistence, etc.

Chapter 4 is devoted to a custom hardware solution for minimizing hardware cost on RTLS systems implemented with RFID hardware, by allowing the deployed RFID readers (quite costly) to be connected to up to 16 antennas in a multiplexed manner.

In Chapter 5 we present the numerical models of three antennas made up of ECPs that cover different frequency bands, compatible with both WiFi and WiMAX standards. We also propose different numerical design techniques (for each antenna) which allow them to be straightforwardly adapted to all kind of frequency ranges. Finally, we constructed experimental prototypes for the first and third antenna model, and after measuring them in an anechoic chamber we compared their performance and parameters with those obtained by simulation.

Finally, Chapter 6 is devoted to conclusions and future lines of work.

Chapter 2

Single-Technology Location Systems

In this chapter we introduce a first approach to a distributed multi-layer software architecture for implementing single-technology Real-Time Location System (RTLS), in the particular case of using wireless technologies based on Received Signal Strength (RSS) information. This chapter is partly based on the publications [65, 128–130].

The software architecture described in Section 2.1 is based on well known solutions from the software engineering discipline, which are introduced in Section 2.1.1. In Section 2.1.2 we propose a possible implementation using an eXtensible Markup Language Remote Procedure Call (XML-RPC) standard protocol, for isolating and distributing the location algorithms from the Wireless Sensor Network (WSN) hardware. In Section 2.1.3 we explain how to access the hardware remotely from a high-level simulation environment such as Matlab. In Section 2.2 we explain two real implementations of the physical layer abstraction, for the particular cases of Bluetooth Classic and ZigBee wireless sensor networks, both based on RSS measurements. In Sections 2.2.2 and 2.2.4 we show some additional hardware implementations and techniques based on the previously defined physical layers, and some experimental results based on real WSN deployments using Bluetooth Classic and ZigBee networks, in this case explaining how the measurement campaigns were made and some tracking results were obtained. Finally, Section 2.3 is devoted to the conclusions.

2.1 Distributed Multi-Layer Software Architecture for RSS based RTLSs

2.1.1 Introduction

Before continuing, we introduce the following software engineering concepts:

- **Software architecture:** This concept was first presented by Edsger Dijkstra in 1968 as a result of his research work. Later on, other authors such as Mary Shaw and David Garlan

continued examining ways in which architectural issues could impact software designs, and examining useful abstractions and paradigms of system designs [98, 140]. The software architecture of a computing system models the structures of the system, which comprise software components, the externally visible properties of those components, and the relationships between them. Later on, the concept of *pattern* was introduced by Erich Gamma [97] of the Gang-of-Four (GoF), and was defined as an effective and reusable way to solve a non-trivial software problem. And this was also applied to the software architecture discipline by other authors [79, 80]. As a result, a new concept baptised as *architectural patterns* appeared.

- **Software layer:** Also known as layer, tier (or software tier), it was generalized in [79] by means of the so-called *layers architectural pattern*. Basically, a layer is a piece of software implementing a clearly defined set of functionality and access interface. Using this layer concept multiple advantages arise when applied to the design of software architecture:
  - One layer hides the implementation technology used from other layers in the system.
  - They can be reused in other systems.
  - They can be physically separated from each other in a distributed manner.
- **Distributed system**. Software systems are distributed when several parts (maybe layers) that compound the whole system are physically separated, among several processors or machines, usually connected through a communication channel (wired or wireless). This also provides the opportunity to process different parts of the system in parallel using several processors, Digital Signal Processor (DSP), Field Programmable Gate Array (FPGA), micro-controllers, etc.

According to the concepts defined above, we have designed and implemented a distributed multi-layer software architecture suitable for real-time localization.

We can start dividing any localization system mainly in two layers, as shown in Figure 2.1:

- **Location:** This layer implements a location algorithm that is fed with data from the bottom layer.
- **Physical:** This layer is responsible for abstracting the way we access the WSN hardware to collect the measurements.

Both layers are independent and reusable, so we can change the wireless technology used in the WSN without changing the *Location* layer, or change the location algorithm without altering the *Physical* layer.

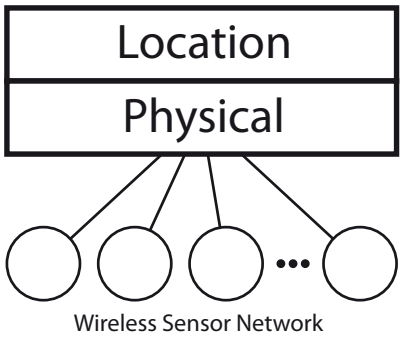


Figure 2.1: Minimum layer separation of any location system.

To separate these two layers in a distributed manner we can communicate them by using different standard mechanisms. And in this manner, we are able to execute both layers in different machines.

One possibility could be to implement a client-server architecture with a star topology by means of one centralized server, abstracting the calls between the *Location* algorithm clients (consuming measurements from the server) and several *Physical* clients (providing measurements to the server), and typically managing a database for storing the measurements.

Another simpler solution is to implement a client-server architecture without a centralized server side, by using a Remote Procedure Call (RPC) protocol, which allows the *Location* layer to remotely call subroutines or procedures of the *Physical* layer, by using a shared network.

When choosing this second RPC mechanism we have several possibilities, such as XML-RPC [61] (introduced in 1998 by Microsoft, which uses XML to encode its calls and Hypertext Transfer Protocol (HTTP) as a transport mechanism), its successor Simple Object Access Protocol (SOAP) [54] (also introduced by Microsoft as an extension of XML-RPC), or the JavaScript Object Notation Remote Procedure Call (JSON-RPC) [50] protocol which is similar to XML-RPC but encoding the data in JSON format instead of in XML.

At the time of implementing our first distributed multi-layer architecture, we decided to use the first XML-RPC option, due to the following advantages:

- It is an open specification standard, with really light and mature implementations available in all kind of programming languages (Python, C and C++, Cocoa, Erlang, Groovy, Java, JavaScript, PHP, Perl and others) and operating systems (multi-platform). Therefore, it is possible to implement the *Physical* layer (which accesses the hardware) using low-level languages and libraries (typically programmed in C), while we can choose another higher-level programming language (Java, C++, Matlab, etc.) to implement the *Location* layer.

- Both layers can run on a single machine or on different distributed machines, interconnected by a shared network or the Internet. In this case the data is XML encoded

and is transported using the HTTP protocol, so that it passes through any firewall without opening any port.

- As a client-server architecture, we chose the *Physical* layer to be the server side, allowing concurrent access to WSN measurements from multiple *Location* clients. These location clients can implement different location algorithms based on the same measurements kind of data, which can be processed and presented at different levels.

2.1.2 XML-RPC Architecture Implementation

In Figure 2.1 in the previous section, we showed the typical structure of a non reusable location system implementation, where the code to access the WSN hardware and the code for implementing a location algorithm are at the same level, or separated but really constrained together in the same machine.

Instead, we have proposed the distributed four-layer architecture shown in Figure 2.2 based on a RPC client-server approach, by using XML-RPC in particular for its implementation.

To separate the *Location* and *Physical* layers (previously described) and communicate them using XML-RPC, we added a communication abstraction layer called *Wrapper* (in the *client*

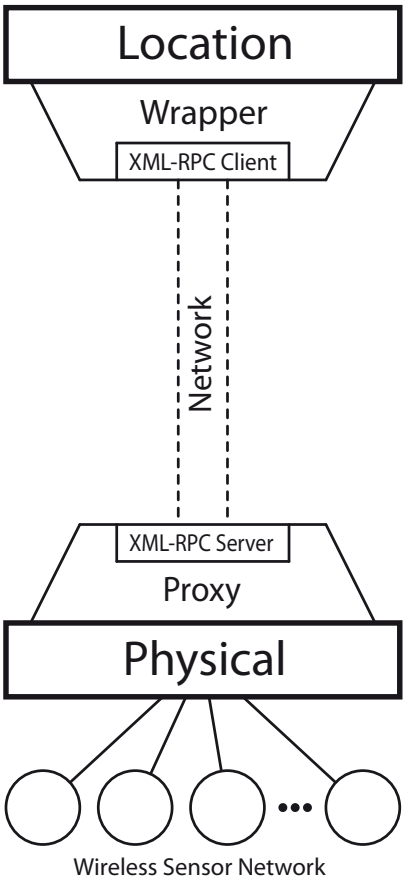


Figure 2.2: Architecture proposal.

side) below the top *Location* layer, and a communication abstraction layer called *Proxy* (in the *server* side) above the bottom *Physical* layer. This way we make any of the two main layers reusable and independent, so that they can be implemented by using different programming languages, and deployed in the same or in different machines. Fortunately, at the time of implementing this architecture there were 79 open implementations [58] of both XML-RPC clients and servers, available for all major programming languages (Java [42], C [62][59], C++ [59], PHP [62][60], etc.). All of them can be used without any restriction.

We will now explain the four layers of the raised architecture in further detail.

Physical Layer

The *Physical* layer as described in this chapter only supports a subset of location systems, based on a WSN of anchor nodes (with fixed and known positions). The anchors capture small packets from which to extract RSS information, with the main objective of ranging several mobile nodes.

This layer is responsible for communicating with the WSN physically and for giving format to gathered measurements in a standard manner that the *Proxy* layer can understand. Depending on the wireless technology used (i.e. Bluetooth, ZigBee, WiFi, etc.), a different interface should be used to communicate to the anchor hardware from the Host computer. The Host should capture in real-time the RSS measurements, typically by means of an Received Signal Strength Indicator (RSSI) (which depending on the hardware manufacturer can be converted into an appropriate received signal level) and provide this information to the upper layers. It is usually implemented at a low level in languages such as C, by using operating system libraries, or by directly communicating with the hardware nodes to a serial port, typically a UART or a USB virtual COM port, etc. This is always specific to the hardware type of the WSN.

The main purpose of this layer is to totally abstract the WSN technology to the *Proxy* layer in charge of communicating the RSS measurements to the upper layers. And this layer can now be replaced by another that accesses different WSN hardware technology, without any further changes in the other layers.

To this end we should implement a program, defined as *port*, for each hardware technology chosen to implement a WSN. In Section 2.2 we will show two example implementations of this *port* for the cases of Bluetooth Classic and ZigBee technologies.

In our case, we fixed some particular restrictions that every *port* implememtation should meet in order to guarantee the compatibility with its successor upper *Proxy* layer:

- It must be possible to launch the *port* from command line (without GUI), and configure it by adding several arguments. Thus, parameters such as anchor identifiers, Media Access Control (MAC) addresses, UART or USB ports, etc., can be configured just by passing

them though arguments to this piece of software. All the parameter depend on the WSN technology in particular.

- It must gracefully stop when receiving a *sigint* [142] signal.
- All RSS data gathered from the WSN must be sent to the *standard output* (stdout) immediately and in real time, using the following standardized format that the *Proxy* layer understands (using a blank space as field separator, and in ASCII format):

  `Timestamp TransmitterID ReceiverID RSSI`

  where the *Timestamp* is the current time epoch (in milliseconds) in UNIX format [57] of the Host, the *TransmitterID* is the mobile node identifier, the *ReceiverID* is the anchor identifier, and the RSSI is the received signal strength indicator.

Proxy Layer

It is implemented in the *server* side over the *Physical* layer. It can be implemented in a programming language that is different to the one used by the *Physical* layer, but the same as the XML-RPC server side library used.

This layer maintains a circular buffer with the last N measurements for each pair (TransmitterID, ReceiverID), storing the timestamp (or instant of time) when each measurement was received. It implements an XML-RPC server to provide the following procedures that may be remotely invoked from multiple XML-RPC clients (aka. *Location* algorithms).

This *Proxy* layer supports the following remote calls:

- **Get RSSI buffer by ReceiverID:** It returns a list of the latest RSS measurements that have been observed by an anchor node identified by `ReceiverID`.
- **Get RSSI averaged measurements by ReceiverID and TimestampRef:** This procedure is similar to the previous one, but it returns a list of RSS measurements averaged by `TransmitterID` and filtered by a time reference `TimestampRef`, to only return the averaged RSS by mobile node after that time reference.
- **Clean RSSI buffer by ReceiverID:** It cleans the measurement buffer of a given anchor identified by `ReceiverID`.
- **Start the Physical layer:** It starts the *Physical* layer by launching the specific *port* process which is responsible for communicating at low-level with the WSN.
- **Stop the Physical layer:** It stops the *Physical* layer by sending a *sigint* signal to the *port* process responsible for communicating with the WSN. In this case, all the RSSI buffers stored for the anchor nodes remain intact and are not cleaned. These buffers will be increased again when a ”Start the Physical layer” is received.

This layer is the most complex layer of the entire architecture because it implements an XML-RPC server which provides the five previous procedures, and it maintains certain data structures with measurement buffers, etc.

So, before starting to describe all the different parts that make up the *Proxy* layer in depth, we must explain the data structures used to store the various RSS and Timestamp buffers for the different mobile nodes detected by several anchors of the WSN. As shown in Figure 2.3 the *Proxy* maintains a hash table which is indexed by the anchor node identifiers. For each entry in this table it hangs a list of the mobile devices that were detected by each anchor. And in turn, for each discovered device it must handle another two lists, which will store (Timestamp, RSSI) tuples of all observed measurements.

Figure 2.3 shows a simplified example of this data structure, where data is shown only for the first two anchors of a WSN and in turn only for the first two mobile devices detected by one of the anchors. These lists will grow automatically as new devices are detected or as new measurements of existing mobile devices are observed, up to a certain RSSI buffer limit (defined by a `maxData` variable in the *Proxy* code). If this limit is exceeded, new measurements will be stored according to a chronologically ordered FIFO queue, where new elements are added at the end and the oldest ones are removed from the beginning. This way, we always have available the last `maxData` measurements for each detected mobile node, and we can fix this variable to a certain high value depending on the Host RAM memory available.

Note that in this specific architecture implementation we have decided not to make the measurements to a database persistent, but this feature could be supported without much effort to the *Proxy* implementation, by inserting each new received RSS measurement into the DB when it is received.

Figure 2.4 shows a simplified UML class diagram of the *Proxy* layer, which represents the classes that form the *Proxy* layer as well as the relationships and dependencies among them.

We have implemented this layer by using the specific Apache *XMLRPC* [42] *server* implementation due to its simplicity of use, open-source code, excellent documentation and because it is implementated in Java, a powerful object-oriented language.

While the *Physical* layer is normally implemented using a low-level language (such as C), we instead used Java to implement the *Proxy* layer. As said before, there is no restriction in choosing different programming languages to implement different layers.

The signature of the procedures that can be remotely requested to the *Proxy*, as well as their parameters, their data type and type of the returned data, are defined in the interface `Facade`, shown on top of the previous UML diagram. This facade represents the XML-RPC server `requestProcessor` that basically defines the signatures of the procedures published by the class `MyXmlRpcServlet`. This class is really simple because it inherits almost full functionality of the class `org.apache.xmlrpc.XmlRpcServlet`, which actually implements the XML-RPC server, which is already implemented by the Apache team.

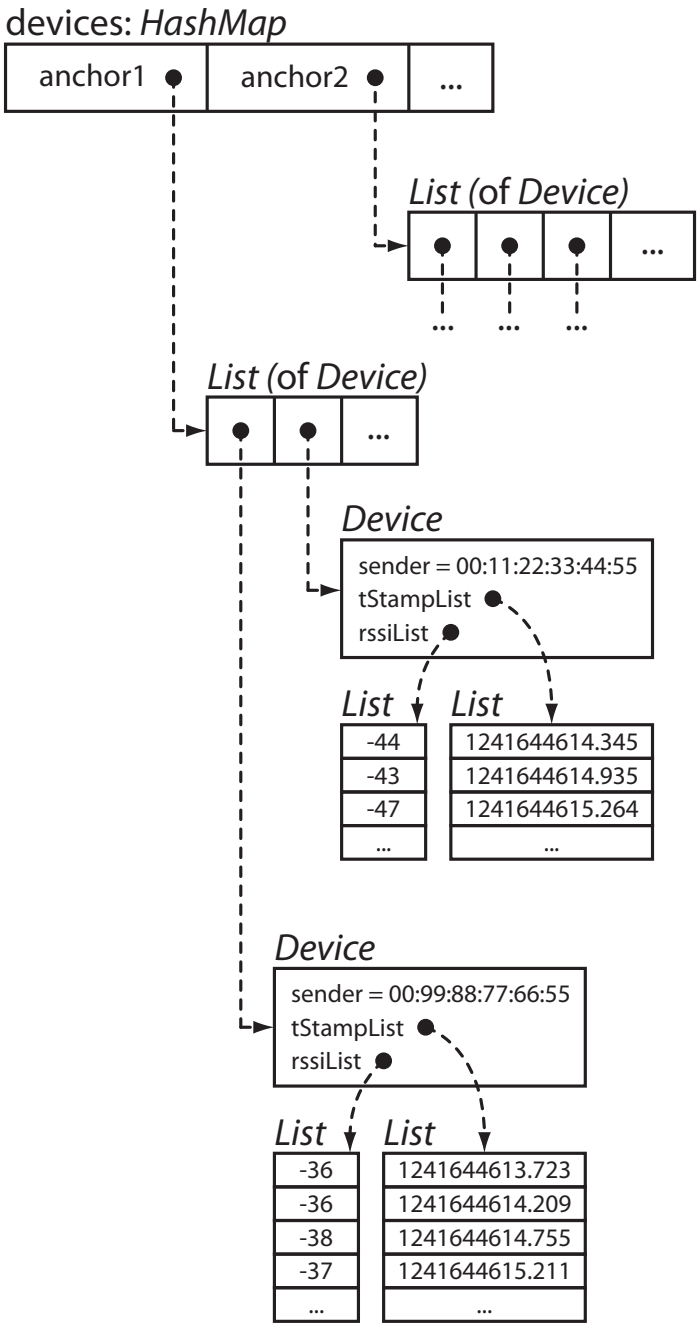


Figure 2.3: Main Proxy layer data structures.

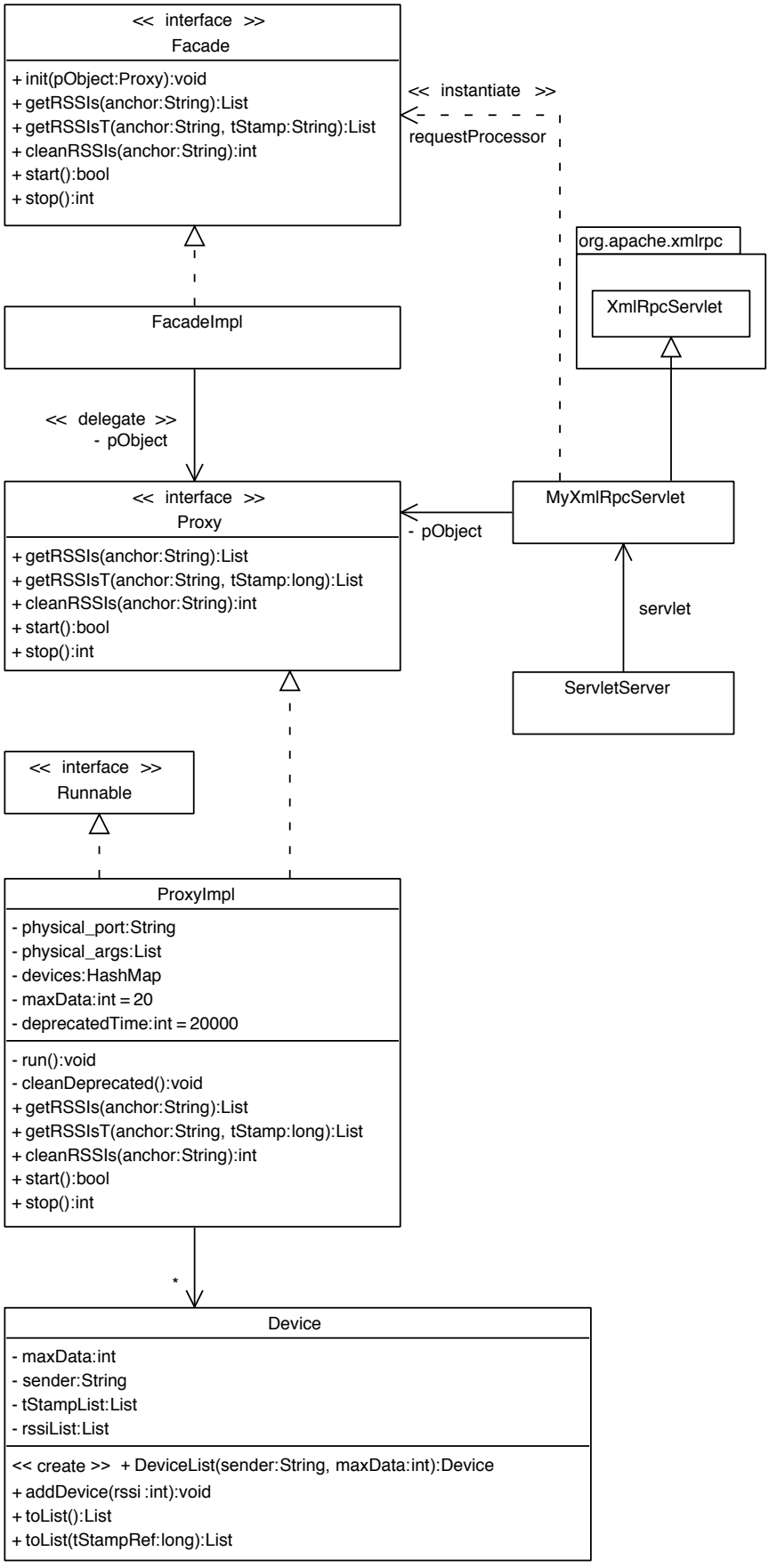


Figure 2.4: UML Class-Diagram of the Proxy layer.

To start this server (*Proxy* layer), we should just run the class *ServletServer* from the command-line, passing as parameters the path to the *Physical* layer *port* and other configuration arguments such as the anchor node identifiers or the UART/USB port used to actually access the hardware. This *ServletServer* class creates a `servlet` of type `MyXmlRpcServlet` and passes it these configuration arguments. Then, it instantiates a `Proxy` object (`pObject`) which is set within the *Facade* by the method `init`. Any remote request will be answered by the servlet, which will directly delegate into this `pObject`, method by method. For this reason the interface `Proxy` has the same methods as the interface `Facade` (except for the `init` one). An additional advantage of this delegation is the capability to rewrite the XML-RPC procedure parameters when needed, to another that is more appropriate for the programming language used (for instance transforming numbers to strings, and vice versa, if needed).

We decided to use threads to implement the `Proxy` interface, and this is why `ProxyImpl` implements `Runnable`. When the `start()` procedure is remotely invoked, a new thread is created, which executes the *Physical* layer *port* process and attends this standard output continuously. So, it remains in an infinite loop, continuously parsing everything the *port* writes to the *stdout*. As soon as new measurements are detected, the thread updates the `device` hash table (inside `ProxyImpl`) and the Timestamp and RSSI tables of each of the `Device` objects which hang from each anchor in the hash table.

The `stop()` procedure sends a *sigint* signal to the *Physical* layer *port* and then terminates the corresponding thread.

The `getRSSIs`, `getRSSIsT` and `cleanRSSIs` procedures have an `anchor` parameter that represents its identifier. They use this identifier to access the `device` hash table, to obtain or clean the corresponding measurement list. The `getRSSIsT` procedure is very similar to `getRSSIs` but it averages the RSSI measurements by `TransmitterID`, which is filtered by a specific timestamp reference instant.

Wrapper Layer

This layer is much more simple than the previous one, and is implemented in the *client* side below the *Location* layer. It implements a XML-RPC client side and simply abstracts all the XML-RPC request logic to the *Location* top layer. It can be implemented using a different language than all other layers, but matching the language of the XML-RPC *client* library used. It also abstracts to the higher layer the way of collecting data that is received in streaming mode. This greatly eases its adaptation to process incoming real measurements, which would arrive in real-time from a WSN.

It supports the 5 procedures explained in section 2.1.2, and there are two different ways of implementing this client:

- Creating an `XmlRpcClient` instance and directly calling its `execute` method, passing

a `methodName` and `parameters`.

- Using a *transport factory* [42] to create `Facade` instance, and then calling one of the 5 methods directly from the programming language.

For each of the 5 procedures supported by the architecture, the *Wrapper* should simply implement a method that "wrappers" it, hiding all the XML-RPC logic from the *Location* layer above it. This way we abstract the top location layer from the XML-RPC request logic, the parameters involved in the request and the way of getting the remote measurements.

A example implementation of a Java Wrapper, called directly from Matlab, is shown in a code view (Figure 2.5) in Section 2.1.3, which is trivially accomplished using the XML-RPC client provided in the Apache XML-RPC framework.

Location Layer

This is the highest level layer which implements a location algorithm, and it uses the *Wrapper* client to transparently obtain experimental measurements.

At the end of the following Section 2.1.3 we show a Matlab sample code which gives us an overview of the steps that should be followed to periodically request for experimental RSS measurements.

2.1.3 Accessing the WSN from Matlab

We have decided to implement the *Wrapper* layer in Java as it can be directly used from Matlab, the most commonly used and the highest-level simulation environment for implementing location algorithms. This Matlab-Java interaction is possible because of the intrinsic connection between Matlab and Java Virtual Machine (JVM), where Java objects can be instantiated transparently, and data type conversions between Matlab and Java types are automatically handled.

To create an XML-RPC Java *client* object by using the client libraries provided by Apache, we should simply create an `XmlRpcClient` object, call its `setConfig` method with our desired configuration parameters (the server side URL or IP address, userName, password, timeouts, etc.), and finally pass this client as a parameter to a `ClientFactory` to directly get an instance of our `Facade`. The local procedure calls to this new facade object will take place on the *Proxy* server in a completely transparent manner. However, there is a simpler alternative to creating this `Facade`, namely by directly using the `execute` single method of the *client* object. We can just specify a remote procedure name (and its parameters) to this generic `execute` function. We have used this method in our implementation, and in the examples shown.

xmlrpc_INIT.m

```
function xmlrpc_INIT()    % Initialize the Java classpath
clear java;
javaaddpath({
 '/XML-RPC_Apache_libs/xmlrpc-client-3.1.jar', '/XML-RPC_Apache_libs/xmlrpc-common-3.1.jar', ...
 '/XML-RPC_Apache_libs/ws-commons-util-1.0.2.jar', '/Proxy_layer_impl/server.jar'});
```

xmlrpc_CONNECT.m

```
function client = xmlrpc_CONNECT(url, port)
import org.apache.xmlrpc.client.XmlRpcClient;
import org.apache.xmlrpc.client.XmlRpcClientConfigImpl;

addr = strcat('http://',url,':',num2str(port),'/xmlrpc');
config = XmlRpcClientConfigImpl();
config.setServerURL(java.net.URL(addr));
config.setBasicUserName('useruser');
config.setBasicPassword('passwordpassword');
config.setEnabledForExtensions(1);
client = XmlRpcClient();
client.setConfig(config);
```

xmlrpc_start.m

```
function xmlrpc_start(client)
client.execute('server.Facade.start', {});
```

xmlrpc_stop.m

```
function xmlrpc_stop(client)
client.execute('server.Facade.stop', {});
```

xmlrpc_getRSSIs.m

```
function rssi = xmlrpc_getRSSIs(client, anchor)
rssi = client.execute('server.Facade.getRSSIs', anchor);
```

xmlrpc_getRSSIsT.m

```
function rssi = xmlrpc_getRSSIsT(client, anchor, tStampRef)
rssi = client.execute('server.Facade.getRSSIsT', {anchor, tStampRef});
```

xmlrpc_cleanRSSIs.m

```
function xmlrpc_cleanRSSIs(client)
client.execute('server.Facade.cleanRSSIs', {});
```

Figure 2.5: Wrapper layer Java implementation adapted to Matlab

Summarizing, the evolution of any location algorithm usually starts with a Matlab implementation tested with simulation data. Then, as soon the algorithm is proven, it is usually desirable to test it with experimental data collected in real-time from a WSN, and we can achieve this by using the small and simple Matlab files shown in Figure 2.5. These files directly uses

the XML-RPC *client* library provided by the Apache Team, and compound our *Wrapper* Java implementation.

All the configuration parameters should be first and one time defined in the `xmlrpc_CONNECT.m` file before running the other parts of code.

Figures 2.6 and 2.7 show a sample code and its output, which represents the way of retrieving the list of measurements of an anchor node from Matlab. All the logic is performed by using the seven ".m" `Wrapper` files shown in Figure 2.5. In this specific example the real measurement data came from a real Bluetooth WSN.

As shown in Figure 2.7, we remotely start the Bluetooth `port` of the *anchor1* after cleaning its measurement list, then we wait for one second to give the *Proxy* time to gather several RSS measurements, and finally we stop the *Physical* layer. In this example we only detected 2 mobile devices (with MACs `00:80:5A:46:7E:DD` and `00:80:5A:46:97:BB` respectively) during a 1 second scan, so that the RSSI and Timestamp lists are quite short: 3 measurements for the first node and 4 for the second one.

At the end of the example, it is also shown how the `getRSSIsT` procedure is called, and how it returns the RSSI data, averaged and filtered by the *tStampRef* instant of time. In this case, as this reference of time is older than the gathered measurement timestamps, all the measurements are averaged, but we could choose an intermediate and older timestamp during a longer measurement campaign to average part of the measurement log. We also receive a $n\_rssi = 3$ parameter, indicating that three measurements were averaged. This parameter is quite important for certain probabilistic tracking algorithms which take the measurement distribution variance into account.

Summarizing, a real location algorithm would perform the following steps to process real measurements. It would enter in a continuous loop doing something very similar to the code in Figure 2.6, which we summarize in the following lines:

1. Sleeps or waits for a certain period of time equal to the location algorithm sample period (i.e. $T_s = 1\,\mathrm{s}$).

2. Get the current system time: $t_0 = now()$.

3. Read measurements from the anchor nodes using `getRSSIsT(anchorID, tStampRef)`, when $tStampRef = t_0 - T_s$.

4. Parse the measurement data within the returned list.

5. Update the location algorithm with the current measurements.

Note that we assume that the time it takes to the algorithm to update is much lower than T_s.

```
% Long output format to avoid exponential notation.
format long g;

% Initialize the Java classpath
xmlrpc_INIT();

% Get the current system time in milliseconds as timestamp
tStampRef = java.util.Calendar.getInstance.getTimeInMillis

% Connect to the Proxy layer
url = '127.0.0.1';
port = '8001';
c = xmlrpc_CONNECT(url, port);

% Clean the RSSIs and Timestamp lists of the anchor1
xmlrpc_cleanRSSIs(c, java.lang.String(anchor1));

% Starts the Bluetooth Physical layer
xmlrpc_start(c);

fprintf('--> Starting...\n');
fprintf('--> Waiting 1 second while the RSSI lists grow\n');
pause(1);

% Get the measurement list
rssis = xmlrpc_getRSSIs(c, java.lang.String(anchor1))

% Stop the Physical layer
xmlrpc_stop(c);
fprintf('--> Stopping...\n');

device1 = rssis(1)    % Point to the first detected device
device2 = rssis(2)    % Point to the second detected device

% Get the chained sub list of measurements of the first device
device1_List = device1(2)

% Get the measurement list averaged and filtered by tStamp
rssisT = xmlrpc_getRSSIsT(c, java.lang.String(anchor1), java.lang.String(tStampRef))

% Point to the first detected device
% Same info as device1 but averaged and filtered by tStamp
device1T = rssisT(1)
```

Figure 2.6: Matlab Wrapper sample code for obtaining some experimental RSS data

```
tStampRef =
             1241644613812

--> Starting...
--> Waiting 1 second while the RSSI lists grow

rssis =
java.lang.Object[]:
    [2x1 java.lang.Object[]]
    [2x1 java.lang.Object[]]

--> Stopping...

device1 =
java.lang.Object[]:
    '00:80:5A:46:7E:DD'
    [6x1 java.lang.Object[]]

device2 =
java.lang.Object[]:
    '00:80:5A:46:97:BB'
    [8x1 java.lang.Object[]]

device1_List =
java.lang.Object[]:
    '1241644614045'
    '-46'
    '1241644614596'
    '-47'
    '1241644614710'
    '-47'

rssisT =
java.lang.Object[]:
    [4x1 java.lang.Object[]]
    [4x1 java.lang.Object[]]

device1T =
java.lang.Object[]:
    '00:80:5A:46:7E:DD'
    '1241644613812'
    '-46.6666666666667'
    '3'
```

Figure 2.7: Matlab Wrapper sample code output

2.2 Physical Layer Implementations and Experiments

As we introduced in the foregoing Section 2.1.2, to implement a *Physical* layer we only need to implement a *port* program able to provide to the *Location* layer a standardized interface and data formatting (through a transparent communication channel provided by the other two *Proxy* and *Wrapper* layers). These *port* requirements can be summarized as:

- A command line tool able to receive some configuration parameters such as an argument (e.g. communication port IDs, anchor ID, etc.).
- All RSS data gathered from the WSN must be sent to the *standard output* (stdout) using the following ASCII formatting `Timestamp TransmitterID ReceiverID RSSI`.
- It must gracefully stop when receiving a *sigint* signal.

In this section we will show two different *Physical* layer implementations using Bluetooth Classic and ZigBee hardware, and for each one some experimental results or additional contributions.

2.2.1 Bluetooth Classic Overview

Bluetooth is a wireless protocol based on the IEEE 802.15.1 standard that allows short range communications between electronic devices, creating Personal Area Networks (PANs). Originally it was conceived to replace RS-232 data cables wirelessly, allowing connection between multiple devices at the same time and solving the synchronization problem among them.

Since the 4.0 Low-Energy variant of Bluetooth arrived, all previous Bluetooth standards are considered *classic*.

It works in the free Industrial, Scientific and Medical (ISM) frequency band of 2.45 GHz which is shared with other wireless technologies such as WiFi. This is why Bluetooth implements a radio technology called frequency-hopping spread spectrum, to avoid interferences. It divides the transmitted data into packets, and transmits each packet on one of the 79 designated Bluetooth channels. Each channel has a bandwidth of 1 MHz, while Bluetooth 4.0 uses 2 MHz spacing, which accommodates 40 channels.

In Bluetooth Classic the devices can be classified into three classes (1, 2 or 3) depending on their transmit power (100 mW, 2.5 mW or 1 mW respectively), handling a coverage range of about 100, 10 or 1 meters. Most classic mobile phones which implemented this classic standard were classified as Class-2, with a typical coverage of about 10 meters indoors. This was a quite restricting range when this kind of devices are used for localization.

The first norm of Bluetooth 1.0 was introduced in 1998, was very immature and with great interoperability problems. Soon, in 2002 the 1.1 norm appeared, with several improvements, and it was possible for the first time to obtain RSSI levels, which is the physical parameter that any RTLS system using this technology should use. The Bluetooth narrowband does not allow extraction of other kinds of ranging information such as Time-of-Arrival (ToA), due to the lack of bandwidth.

In 2003 the 1.2 norm appeared, and it incorporated great improvements such as the Adaptive Frequency Hopping (AFH) spread spectrum, which improves the radio interferences with occupied channels, allowing those ”problematic” channels to be avoided automatically. Moreover, a new mechanism to obtain the RSSI level also appeared. For the first time, it was possible to get these levels without establishing and maintaining a network connection between two devices, which really eases the way of implementing a ranging mechanism in an RTLS. This was called *Inquiry*, and made it possible to obtain not only the MAC address (physical and unique address) of the detected device, but also the clock offset and even the RSSI level of the last received packet.

One year later, the 2.0 norm of Bluetooth was defined but it was not until 2005 that it was really integrated in commercial devices. After that time, almost every mobile phone, Personal Digital Assistant (PDA), Personal Navigation Assistant (PNA), laptop, etc. integrate Bluetooth 2.0 connectivity, which was the most commonly used version. The main contribution of this standard was the Enhanced Data Rate (EDR) functionality, which allowed transfer speeds of up to 2.1 Mbps, previously limited to 720 kbps.

In 2007 the 2.1 norm appeared and incorporated an Extended Inquiry Response, which for the first time made it possible to better filter which devices respond to each Inquiry. After this, we received additional information about the device name, list of supported services, time of day and other pairing information.

In 2009 a new 3.0 norm appeared incorporating new PHY/MAC alternatives which allow transfer speeds up to 24 Mbps, but without much success, because this standard was not included in many devices. They even tried to incorporate a UWB physical layer into this standard but it was suspended in October 2009 by the Bluetooth Special Interest Group (SIG).

Finally, in 2010 the Bluetooth Low-Energy 4.0 (BLE) standard appeared, incorporating a number of changes to the standard, focusing on reducing the power consumption (to ultra-low power running off a coin cell) and easing the software implementations (reducing the layers and complexity of the Bluetooth stack). Previously known as Wibree, BLE defined an entirely new protocol stack for rapid build-up of simple wireless links. Since the first BLE standard was released, a further two 4.1 and 4.2 revisions have appeared as software incremental releases, not hardware updates. They offer more connectivity and topology options, add other secure mechanisms and even provide IPv6 functionality to support connected home and other Internet of Things (IoT) implementations.

The experiments and measurement campaigns in this chapter were performed using Bluetooth Classic 2.0 hardware as described next.

Integrating Bluetooth Classic Hardware with the RTLS Architecture

When we implement a WSN using Bluetooth Classic technology, the network and its operation would look like in Figure 2.8. Each anchor will be connected to the host by USB cable and obtain RSSI data by means of the standarized Bluetooth *Inquiry* [4][96] procedure. The *Physical* layer *port* could easily be implemented maintaining 2 independent processes in memory: one in an infinite loop sending *Inquiry* (0x01|0x0001) Host Controller Interface (HCI) **commands** [4] to the physical Bluetooth device (where only the Bluetooth layers below the HCI are implemented), while the other process parses *Inquiry Result with RSSI* (0x22) HCI **events** [4]. For each parsed event of kind 0x22, the port should just write to its *stdout* a *Timestamp* (the current system time in UNIX format), a *TransmitterID* identifier (mobile node MAC), a *ReceiverID* identifier (anchor MAC) and an *RSSI*.

In Figure 2.8 we show a typical Bluetooth anchor network, where the anchors are represented as circles, and the mobile nodes as squares. All the anchors are connected to a USB HUB by cables that can each be up to 5 meters long (when disabling the USB 2.0 support). They are powered from this HUB and connected to a Host computer, from which a single *port* can by launched to control all of them at once, or by launching N *ports* to attend the measurements received from N anchors, treated individually by each of them.

In our experiments we used Bluetooth 2.0 hardware nodes (shown in Figure 2.9), Aircable [3] Host XR model, with a CSR [5] BlueCore4-Ext chipset. These devices have the advantage of having an external SubMiniature version A (SMA) antenna connector so that they

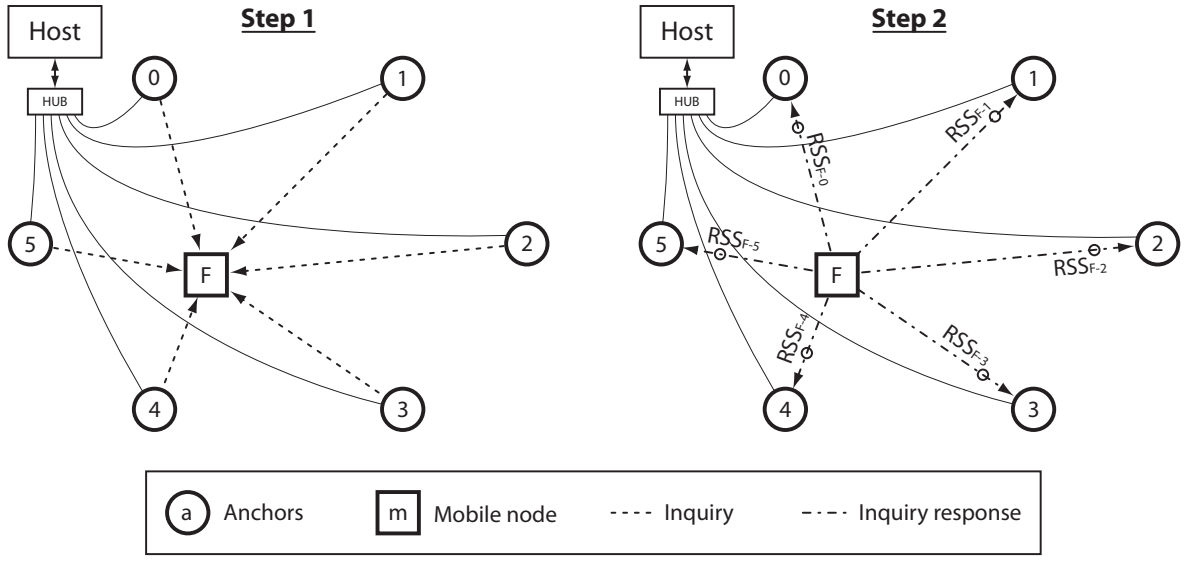


Figure 2.8: Process for getting RSS measurements using a Bluetooth Classic sensor network.

Figure 2.9: Bluetooth Classic industrial node (Aircable Host XR model), used for the implementations.

can be used with different omni-directional antennas, to better adapt to the characteristics of the location scenario, its walls, etc. As we said, since these devices only have the Bluetooth layers below the HCI layer implemented in their firmware, we always need a host computer to send HCI commands to start the Inquiry process, and to configure the hardware scanning window and intervals. And in this Host computer we also receive the generated HCI events from which we extract the RSSI measurements. In the case of the BlueZ [45] Bluetooth stack in GNU/Linux, the RSSI returned measurements are directly mapped to RSS real values, with a resolution of 1 dBm. This process is reflected in the two consecutive steps shown in Figure 2.8, which are repeated continuously.

One of the major problems when implementing a WSN using Bluetooth Classic is the distance limitation due to the USB 2.0 cable restrictions from the anchors to the host. The cables can usually be up to 2-3 meters without repeating the signal with an intermediate USB HUB or an active cable. But, by disabling the USB 2.0 support in the operating system (disabling the `ehci-hcd` support in GNU/Linux, we only support the USB 1.1 standard), we can still communicate at a lower speed (USB fast-speed of 12 Mbps or USB low-speed 1.5 Mbps) to the Anchors, in this case by using up to 5 meter long cables. Note that all USB 2.0 devices are USB 1.1 backward compatible.

This second approach of implementing one *port* per anchor is the one we followed, so that we have to create N wrapper clients from the *Location* layer to receive measurements from all of them individually.

By using a star topology made by a central HUB and some intermediate HUBs we can cover a certainly large room space, hiding the cables on a floating ceiling or below a technical floor (taking into account that the floor tiles would attenuate and quite drastically affect the radiation pattern of the anchor antennas). Nevertheless, both options were quite inexpensive at the time of implementing these RTLS solutions back in 2005 to 2007, and valid for certain scenarios.

2.2.2 Bluetooth Classic Implementations and Experiments

2.2.2.1 Application to Joint Estimation of Position and Channel Propagation Model Parameters in a Bluetooth Network

We applied the proposed architecture for the experimental evaluation of a Bayesian filtering method, for the joint estimation of position and channel propagation model parameters, using a Bluetooth WSN in an indoor scenario [130].

Introduction

WSNs are widely recognized as a solution to implement indoor location systems [105]. These networks are complementary to Global Navigation Satellite Systems (GNSSs) [99], which are only valid for outdoor environments where there is Line of Sight (LOS) with the satellites.

A target node position is estimated by considering anchor nodes with known positions as a reference, and several technologies such as Bluetooth, ZigBee or UWB can be used for ranging purposes using different propagation parameters. In this case we decided to use a Bluetooth WSN and, therefore, the RSSI information is used for ranging.

It is well know that the signal levels of RSSI vary randomly according to the environmental circumstances (presence of obstacles, multipath, people moving around, etc.), and due to these fluctuations the location system accuracy decreases. While these variations can be modelled with small and large scale models [127] that statistically represent the origin and type of these variations, a propagation model should be considered to process the received signal power observations.

In this study we considered the classic path-loss propagation model, based on the principle of signal attenuation over a distance to describe the signal variations versus distance, as follows:

$$P_L(d)[dB] = P_L(d_0) + 10\,n\,\log_{10}\left(\frac{d}{d_0}\right) + X_{\sigma_L} \tag{2.1}$$

where $P_L(d)$ and $P_L(d_0)$ are the power level of the received signal at the distances d and d_0 respectively, d_0 is the reference distance, n is the path-loss exponent and X_{σ_L} represents random noise with a Gaussian distribution, with zero mean and σ_L standard deviation.

Although there are several techniques that try to reduce the measurement variations that appear when using sensor networks [105], if the path-loss exponent n is wrongly assumed these systems do not work correctly. The path-loss exponent value typically varies between 1 and 3 in indoor environments [127] when there is a clear LOS, but it can change suddenly when the line of sight is blocked, i.e. in the case of Non Line of Sight (NLOS). This parameter can empirically be estimated from off-line field measurements at several distances in the environment but, since it is time-variant, a technique to track its value changes is desirable.

In this section we introduce a method for position estimation with dynamic adaptation of this path-loss parameter, when n value is considered between two discrete values, one corresponding to the LOS situation, and another to the NLOS case. This method is implemented using a Bayesian filtering technique [89, 91] that takes into account empirical RSSI measurements from a Bluetooth WSN.

This section is organized as follows. First we present the problem statement, emphasizing the possible changes that could take place in path-loss. Next, we introduce the proposed state-space model and the Sequential Monte Carlo (SMC) method for implementing the joint estimation of position and discrete channel parameters. And finally, we show some results that prove the advantages of the proposed models and algorithm.

Problem Statement

The power attenuation model defined in eq. (2.1) is described by the path-loss exponent n, which should be determined a priori from experimental measurements. The values of n depend on the environment conditions, so if a wrong value is assumed, it will drastically affect the distance estimation and, therefore, the overall accuracy of the position estimation.

We have considered a 6×10 meters indoor scenario with the following experiment: we took real RSSI measurements at several distances (from 1 to 9 meters) between two Bluetooth nodes (mobile target and anchor node), during about four minutes at each position. To observe the difference between the LOS and NLOS cases, we repeated the measurement process by putting an metal obstacle (standard PC enclosure) at 40 cm in front of the target node, to block the main line of sight path.

In Figure 2.10 we show the connection schema of the hardware, and how the measurements were taken, following the nomenclature shown in Section 2.2.1. In this particular case only one anchor was used (AirCable Host XR with a 2 dBi omnidirectional antenna, denoted as 1 in the figure) which was connected to a laptop via USB, whilst at the other end we used a Nokia mobile phone (placed on a box and on a table at the same height as the anchor), acting as a mobile target device.

As explained in detail in Section 2.2.1, in order to obtain the measurements we used the standard Bluetooth discovery capability called *Inquiry*. The anchor node acted as a master, transmitting Inquiry packages, while the mobile phone (target node) acted as a slave, replying to these discovery packages. From the response events we extracted the RSS measurements.

Figure 2.11 shows the received power levels we have observed for the LOS and NLOS cases at several distances between master and slave nodes. The solid green line represents the noiseless model defined in eq. (2.1) when a path-loss exponent, n, deduced from linear regression, is considered. In the NLOS case the main line of sight path was clearly blocked by the obstacle. As shown, the deduced values of the path-loss were $n_{LOS} \simeq 1.8$ and

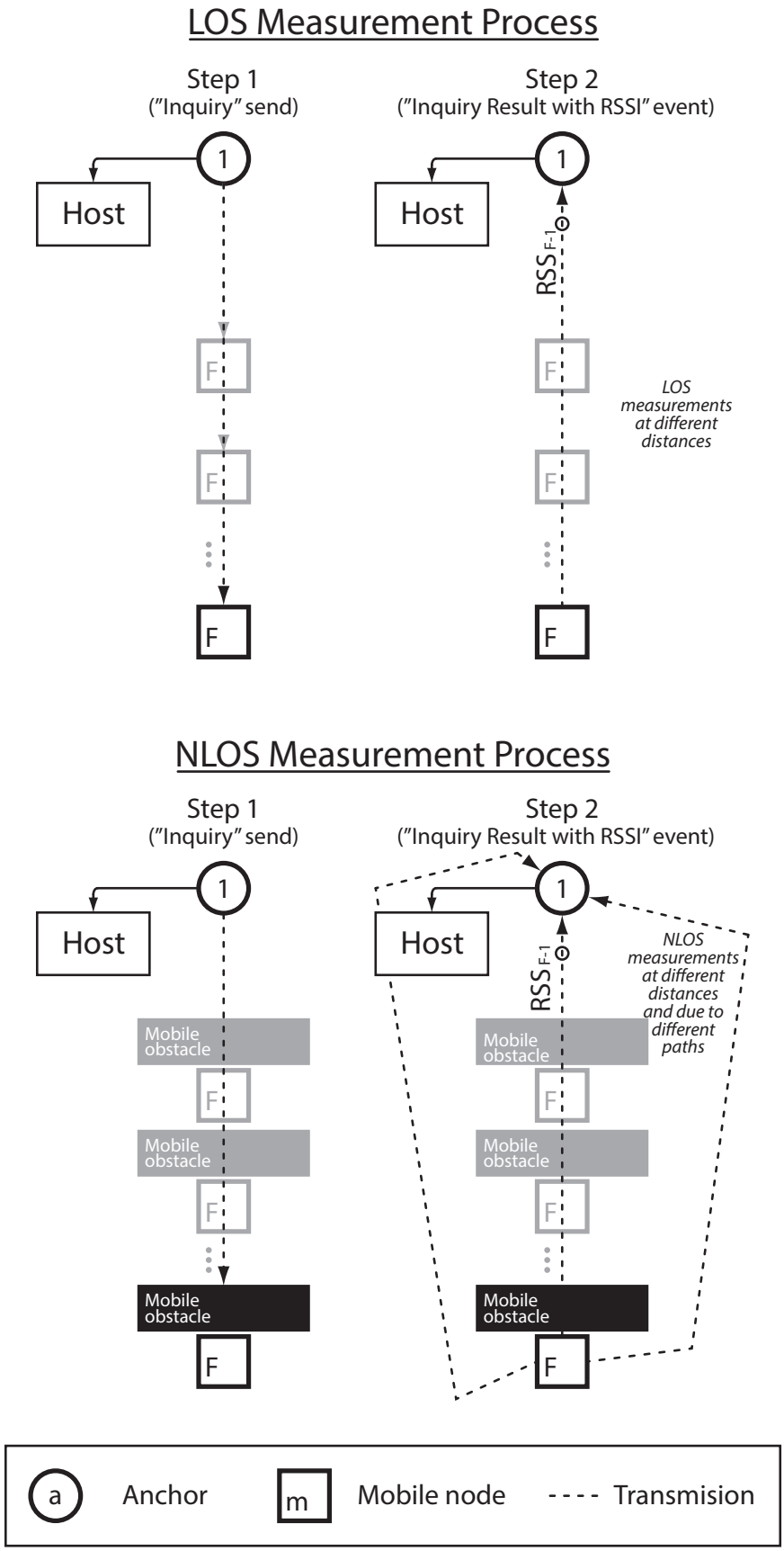


Figure 2.10: Scenario connection schema, and measurement process for the LOS and NLOS cases.

$n_{NLOS} \simeq 0.02$, with $\sigma_L = 5.05$ and $\sigma_L = 5.48$ for LOS and NLOS cases respectively.

Therefore, in order to achieve a reliable location estimation, we must adapt the model to track channel condition changes. There are several works that estimate channel parameters. In [117] the unknown propagation model parameters are deduced from mathematical formulation. The work in [144] proposes parametric propagation models as a feasible way to track the channel. However, these methods are only designed to estimate channel parameters without taking into account the joint estimation with a target node position.

The problem of the joint path-loss exponent and position estimation is very complex because each estimator (one for the position and the other for the path-loss exponent) needs information from the other, thereby making it possible to reach multiple solutions. In order to take into account channel condition variations and guarantee the algorithm convergence, we must reduce the degree of freedom in the path-loss exponent, making it vary only at different discrete values. Our work uses an idea similar to [113] and [119], where the authors considered transitions between different situations (LOS and NLOS) by using a two node Markov model, which takes into account the LOS probability between transmitter and receiver.

Our algorithm considers a target node position estimation and, at the same time, the two conditions of LOS ($n_{LOS} \simeq 1.8$) and NLOS ($n_{NLOS} \simeq 0$), discreetly. In order to achieve this joint estimation we introduce a new algorithm based on a particle filter. A similar idea was used by [123] where they apply these conditions on an Ultra Wide Band (UWB) location system.

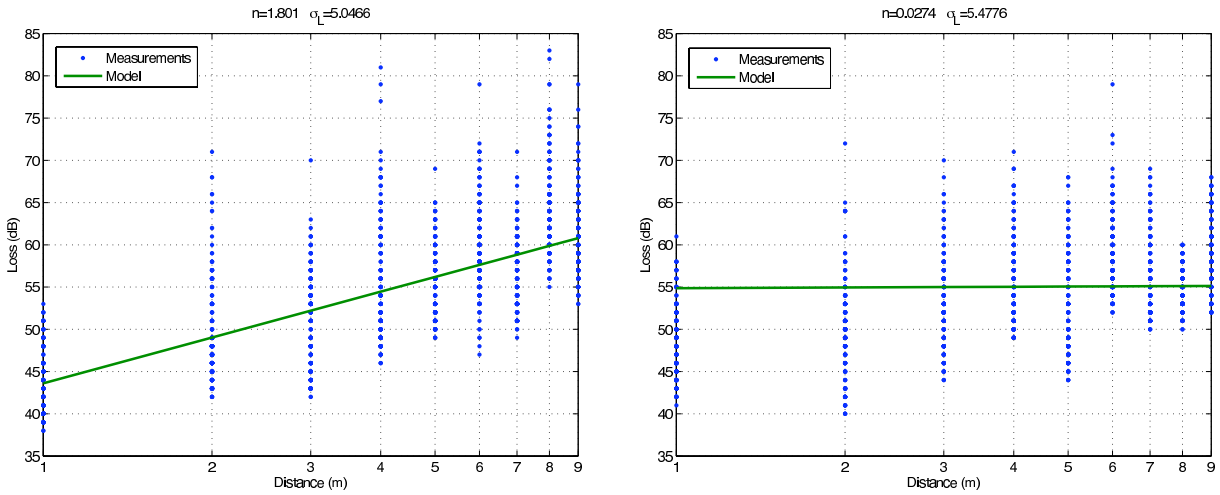


Figure 2.11: Power loss versus distance for the LOS case (left) and NLOS case (right).

Notation

In the following sections we use an argument-wise notation, common in Bayesian analysis. Scalar magnitudes are denoted using regular face letters, e.g., x, while matrices and vectors are written as bold-face upper-case and lower-case letters, respectively, e.g., matrix $\mathbf{X}$ and vector

$\mathbf{x}$. We use the letter p to denote the true probability density function (pdf) of a random variable or vector. For two random variables x and y, $p(x)$ is the true pdf of x and $p(y)$ is the true pdf of y, possibly different. The conditional pdf of x given y is written $p(x|y)$.

State-Space Model: Motion Model

We formally represent the motion of target by means of a Markov stochastic process [102].

To represent the target dynamic state at time $t \in \mathbb{N}$ over a two dimensional region we need at least to estimate the target position $\mathbf{r}_t \in \mathbb{R}$, that is, a 2×1 real vector $\mathbf{x}_t = [\mathbf{r}_t^\top]^\top \in \mathbb{R}^2$.

In this case, for the position estimation we chose the random walk model, whose dynamic equation is described as:

$$\underbrace{\begin{bmatrix} r_{1,t} \\ r_{2,t} \end{bmatrix}}_{\mathbf{x}_t} = \underbrace{\begin{bmatrix} r_{1,t-1} \\ r_{2,t-1} \end{bmatrix}}_{\mathbf{x}_{t-1}} + \underbrace{\begin{bmatrix} u_{1,t} \\ u_{2,t} \end{bmatrix}}_{\mathbf{u}_t} \tag{2.2}$$

where $\mathbf{u}_t$ is 2×1 real-valued Gaussian vector with zero mean and diagonal covariance matrix, $\sigma_u^2 \mathbf{I}_2$, which represents the effect of unknown movements.

As we want to estimate the path-loss exponent and the target position at the same time, we need to add new state variables $n_{j,t} \in \mathbb{R}$ to the previous state variable $\mathbf{x}_t$. We write the full $J \times 1$ path-loss variable vector as $\mathbf{n}_t = [n_{1,t}, \ldots, n_{J,t}]$, where J is the number of anchors for which we want to estimate if they are receiving RSS observations from LOS or NLOS situation.

We assign appropriate transition probabilities for the new introduced state variables $n_{j,t}$:

$$p(n_{j,t} = n_{LOS} | n_{j,t-1} = n_{LOS}) = 1 - \varepsilon, \quad p(n_{j,t} = n_{NLOS} | n_{j,t-1} = n_{NLOS}) = 1 - \varepsilon'$$
$$p(n_{j,t} = n_{NLOS} | n_{j,t-1} = n_{LOS}) = \varepsilon, \qquad p(n_{j,t} = n_{LOS} | n_{j,t-1} = n_{NLOS}) = \varepsilon'$$

where ε is the probability of change to a NLOS situation where the target is currently at a LOS situation (usually a low probability), and ε' is the opposite probability of a sudden change from NLOS to LOS (usually also low, and maybe different from ε). Figure 2.12 summarizes the possible transitions of $n_{j,t}$ using a two-node Markov model.

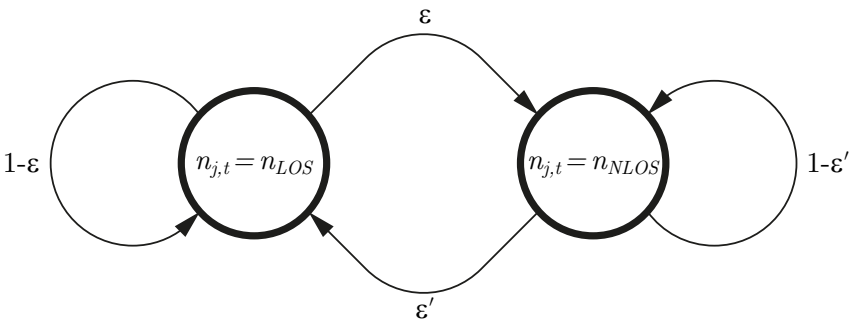


Figure 2.12: Two-node Markov model representing the possible transitions of $n_{j,t}$ based on the probabilities ε and ε'.

Finally we formally represent the target dynamic state at time $t \in \mathbb{N}$ over a two dimensional region as a $(J+2) \times 1$ real vector:

$$\mathbf{x}_{J+2,t} = \begin{bmatrix} r_{1,t} \\ r_{2,t} \\ n_{j,t} \end{bmatrix} \begin{array}{l} \text{x-coordinate in a 2D plane [m]} \\ \text{y-coordinate in a 2D plane [m]} \\ \text{path-loss exponent of the } j\text{-th anchor} \end{array}$$

So, the $(J+2) \times 1$ state vector $\mathbf{x}_{J+2,t} = [\mathbf{r}_t^\top \ \mathbf{n}_{j,t}^\top]^\top \in \mathbb{R}^{J+2}$ evolves with time according to:

$$\begin{aligned} r_{1,t} &\sim p(r_{1,t}|r_{1,t-1}) = N(r_{1,t}; r_{1,t-1}, \sigma^2_{r_{1,t}}) \\ r_{2,t} &\sim p(r_{2,t}|r_{2,t-1}) = N(r_{2,t}; r_{2,t-1}, \sigma^2_{r_{2,t}}) \\ n_{j,t} &\sim p(n_{j,t} = n_{LOS}|n_{j,t-1} = n_{LOS}) = 1 - \varepsilon \\ n_{j,t} &\sim p(n_{j,t} = n_{NLOS}|n_{j,t-1} = n_{LOS}) = \varepsilon \\ n_{j,t} &\sim p(n_{j,t} = n_{NLOS}|n_{j,t-1} = n_{NLOS}) = 1 - \varepsilon' \\ n_{j,t} &\sim p(n_{j,t} = n_{LOS}|n_{j,t-1} = n_{LOS}) = \varepsilon' \end{aligned} \tag{2.3}$$

State-Space Model: Observation Model

We investigate a scheme in where RSS observations are collected from J anchors (or sensors in general). The measurement provided by the j-th sensor at time t is denoted by $y_{j,t}$. In order to describe mathematically the relationship between the observed RSS level, $y_{j,t}$, and the target position, $\mathbf{r}_t$, we use the previously introduced logarithmic path-loss model:

$$\begin{aligned} y_{j,t} &= f_j(\mathbf{r}_t, \varepsilon_t), \\ &= L_0 + 10\, n_{j,t}\, \log_{10}\left(\frac{d_{j,t}}{d_0}\right) + \varepsilon_t \end{aligned} \tag{2.4}$$

where $d_{j,t} = ||\mathbf{r}_t - \mathbf{s}_t||$ is the distance between the position of the j-th anchor and the target at time t, d_0 is the reference distance, L_0 is the path-loss mean value at the reference distance, $n_{j,t}$ is the path-loss exponent of the j-th anchor at time t and, finally, $\varepsilon_t \sim N(\varepsilon_t; 0, \sigma^2_\varepsilon)$ is normally distributed.

We write the full $J \times 1$ observation vector as $\mathbf{y}_t = [y_{1,t}, \ldots, y_{J,t}]^\top$.

Sequential Monte Carlo Method

The smoothing pdf which contains all relevant statistical information for the estimation of $x_{0:t}$ is $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$, which for our non-linear case has no analytical form.

We choose to use a SMC method also known as Particle Filter (PF) [91] to approximate this pdf. The PF generates, in each time instant t, a set $\left\{x_t^{(i)},\ w_t^{(i)}\right\}_{i=1}^{M}$ where $x_t^{(i)}$ is the i-th particle, with weight $w_t^{(i)}$, and M is the number of particles.

Using these particles we can approximate the posterior pdf $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ as

$$p_M(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = \sum_{i=1}^{M} \delta_i(\mathbf{x}_{0:t}) w_t^{(i)}, \tag{2.5}$$

where δ_i is the Dirac delta measure centered at the i-th particle, $x_i^{(i)}$.

Particles can be generated with a variety of proposal distributions as long as the weights are computed accordingly.

If we choose an importance function that can be factorized as

$$\pi(\mathbf{x}_{0:t}^{(i)}) \propto \pi(\mathbf{x}_t^{(i)})\, \pi(\mathbf{x}_{0:t-1}^{(i)}) \tag{2.6}$$

and if we expand the posterior distribution using Bayes′ theorem, we can implement the importance sampling methodology sequentially.

The expression for the weights then becomes

$$\begin{aligned} w_t^{(i)} &\propto \frac{p(\mathbf{x}^{(i)}|\mathbf{y}_{1:t})}{\pi(\mathbf{x}^{(i)})} \\ &\propto \frac{p(\mathbf{y}_t|\mathbf{x}_t^{(i)})\, p(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)})}{\pi(\mathbf{x}_t^{(i)})}\, w_{t-1}^{(i)} \end{aligned} \tag{2.7}$$

A good deal of the performance of the algorithm depends on the choice of the importance function. The simplest choice is to use the prior function.

$$\pi(\mathbf{x}_{0:t}^{(i)}) = p(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)})\, \pi(\mathbf{x}_{0:t-1}^{(i)}) \tag{2.8}$$

So that the previous equation (2.7) can be simplified as

$$w_t^{(i)} \propto p(\mathbf{y}_t|\mathbf{x}_t^{(i)})\, w_{t-1}^{(i)}. \tag{2.9}$$

In Table 2.1 we show a summary of the Sequential Importance Resampling (SIR) particle filter algorithm implementation using the prior as the importance sampling function.

It is well known that the sequential application of 2.3 and 2.9 with a finite number of samples, $M < \infty$, quickly leads to a degenerate set of particles [91]. This problem is commonly overcome with a resampling step which is proven to avoid it [85, 88, 91, 120]. As shown in Table 2.1 the resampling step is performed every time the approximate effective sample size [91] M_{eff} falls below a user-defined threshold. Since $M_{eff} \leq M$, typical threshold values are $\gamma\, M$, with $0 < \gamma < 1$.

Once we have the approximation to the pdf $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$, we can calculate the approximate Minimum Mean Square Estimate (MMSE) of $\mathbf{x}_t$ as

$$\hat{\mathbf{x}}_t^{mmse} = \int \mathbf{x}_t\, p_M(\mathbf{x}_t|\mathbf{y}_{1:t})\, d\mathbf{x}_t = \sum_{i=1}^{M} \mathbf{x}_t^{(i)} w_t^{(i)}. \tag{2.10}$$

Sequential Importance Resampling (SIR) particle filter
(with a prior importance function)

Initialization. We draw M particles $\mathbf{x}_0^{(i)}$ from $p(\mathbf{x}_0)$, and set $w_0^{(i)} = \frac{1}{M},\ i = 1, \ldots, M$.
Recursive steps, for $t > 0$,

1. Draw $\tilde{\mathbf{x}}_t^{(i)}$ from the prior $p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)}),\ i = 1, \ldots, M$, using equation (2.3).
2. Update the importance weights, $\tilde{w}_t^{(i)} = p(\mathbf{y}_t|\tilde{\mathbf{x}}_t^{(i)})\, w_{t-1}^{(i)}$, using (2.9)
3. Normalize the weights, $w_t^{(i)} = \dfrac{\tilde{w}_t^{(i)}}{\sum_{k=1}^{M} \tilde{w}_t^{(k)}},\ i = 1, \ldots, M.$
4. Calculate the effectiveness of the particles, $M_{eff} \approx \dfrac{1}{\sum_{k=1}^{M} w_t^{(k)}}$
5. Perform a multinomial resampling when $M_{eff} < \gamma\, M,\ \gamma \in (0, 1)$. For $i = 1, \ldots, M$, we assign $\mathbf{x}_t^{(i)} = \tilde{\mathbf{x}}_t^{(k)}$ with probability $w_t^{(k)},\ k \in \{1, \ldots, M\}$.

Table 2.1: SIR particle filter algorithm implementation.

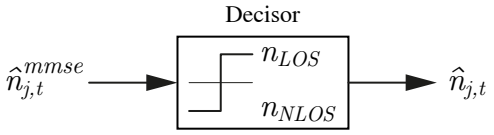


Figure 2.13: MMSE estimator of the path-loss exponent $n_{j,t}$ plus a decisor operation to make it discrete.

Additionally, the MMSE estimator of the path-loss exponent is passed through a decisor operation that compares if the estimator is closer to the n_{LOS} or n_{NLOS} values, and fixes the output to one of these two discrete values $\hat{n}_{j,t} = \{n_{LOS}, n_{NLOS}\} \in \mathbb{R}$, as shown in Figure 2.13. We used an MMSE estimator plus a decisor for the path-loss exponent, instead of a MAP estimator, in order to obtain a good trade-off between computational complexity and estimation accuracy.

Results

In this section we check the performance of the proposed particle filter. As we said before, the algorithm should jointly estimate the position and the path-loss exponent, $n_{j,t}$, for each anchor.

In the following experiments, the possible values of $n_{j,t}$ are limited to detecting LOS ($n_{LOS} = 1.8$) or NLOS ($n_{NLOS} = 0$) cases. These two values are realistic and appeared from experimental data analysis from a real indoor scenario, as shown in Figure 2.11. For all the experiments we also used the following parameters: $\sigma_{r1,t} = \sigma_{r2,t} = 0.3\,\text{m}$, $\gamma = 0.1$,

$\sigma_L = 5$ dBm[1], sampling period $T_s = 1$ s and $M = 1000$ particles.

We simulate 4 anchor nodes at each corner of a 6×10 meter room, which gather RSSI measurements. Then, we tried to locate a target node situated at the $(4, 5)$ coordinate.

1st Experiment: Effect of Choosing a Wrong Fixed Path-Loss Exponent for One Anchor

Before dynamically estimating the path-loss exponent we ran a first experiment that shows the effect of considering a fixed and wrong n value in general, when tracking a target position. In this case the dynamic changes between LOS and NLOS for the path-loss exponent are not considered and we show the effect of consider a shifted n value (due to changing room conditions, obstacles or people moving). To simulate this case, we fixed $n_{j,t} = n_{LOS} = 1.8$ for every particle and made the simulations with a different n_{real} path-loss exponent for one anchor, while the other three anchors use the right and exact path-loss parameter. We show in Figure 2.14 the CDF of the position estimation error when the real path-loss exponent, n_{real}, is different to the fixed value considered. We can see that when the difference between the real and the wrongly considered path-loss exponent value is low (i.e. differences of ± 0.2 around $n_{real} = 1.8$), the algorithm's accuracy is not drastically reduced. And, as soon as the real value changes to an NLOS situation (producing an important n value shift due to an LOS path block), the algorithm obtains highly inaccurate position estimations, shown as dotted lines in Figure 2.14. With this result we can conclude the importance of dynamically adapt the path-loss exponent to different situations with regard to each anchor.

[1]The σ_L considered value is realistic and typically appears in real indoor environments when using Bluetooth classic hardware nodes, as shown in Figure 2.11.

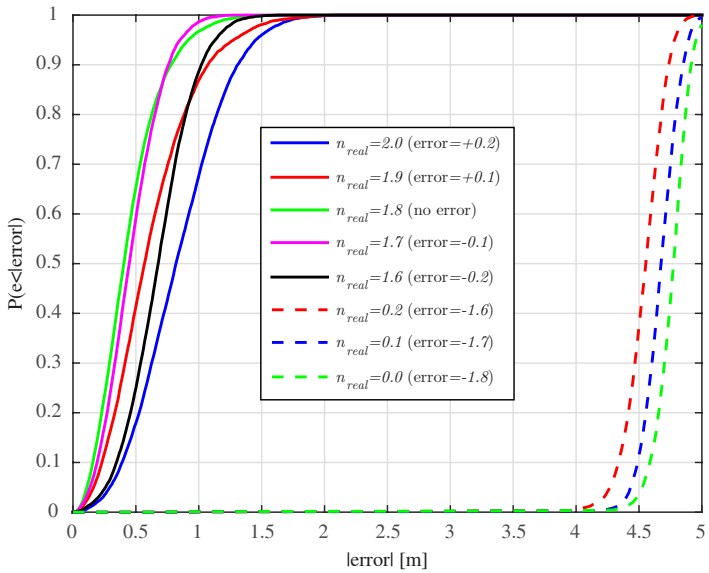


Figure 2.14: Cumulative Distribution Function (CDF) of the position prediction error for 4 anchors in LOS situation, when the $n_{j,t}$ parameter for one j-th anchor is wrongly assumed.

2nd Experiment: Effect of Change from an LOS to NLOS Condition for One Anchor

In a second experiment, the accuracy of the introduced particle filter model is tested when there is a change from LOS to NLOS situation in the middle of the experiment, again for a single anchor, while the other three anchors remain in a correct LOS condition. Figure 2.15 shows the tracking speed after switching from LOS to NLOS at $t = 100$ s. Three different probabilities of change $\varepsilon = \varepsilon'$ values were used (see (2.3) for more details). We can see in the figure that a low ε probability is needed in order to achieve a good $\hat{n}_{j,t}^{mmse}$ estimation, but with lower ε values more iterations are needed to correctly estimate the real n_{real} path-loss exponent. For instance, with $\varepsilon = 0.10$ we switch from these two conditions in just 2 iterations, while when using $\varepsilon = 0.02$ we obtain a better $\hat{n}_{j,t}^{mmse}$ estimation but we converge in about 7 iterations of the algorithm.

So, as soon as we increase ε, we obtain a faster but noisier $\hat{n}_{j,t}^{mmse}$ estimation, closer to the limit zone of the decisor threshold. Therefore, we should take care not to increase ε too much so as to avoid crossing this limit, in order to avoid path-loss estimation errors.

After passing through the decisor, we can see that the final $\hat{n}_{j,t}$ discrete estimation would always be correct for the three ε values shown, so we can safely choose the faster converging one ($\varepsilon = 0.10$). In this experiment we used $n_{threshold} = (n_{LOS} - n_{NLOS})/2 = 0.9$, shown as a pink dashed line in the figure.

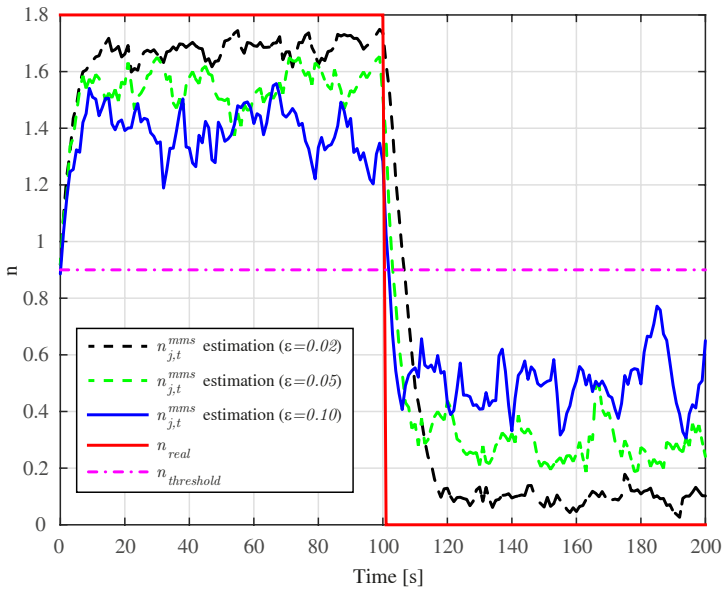


Figure 2.15: Path-loss $n_{j,t}$ discrete prediction to detect the LOS or NLOS case in the changing anchor.

3th Experiment: Accuracy for Different LOS and NLOS Combinations

Finally, figure 2.16 shows the performance of the PF algorithm when the following four conditions are considered: 0 to 3 anchors in NLOS case while the remaining anchor or anchors stay in LOS case.

The results show a location accuracy < 1 m during 85% of the time for the first two cases.

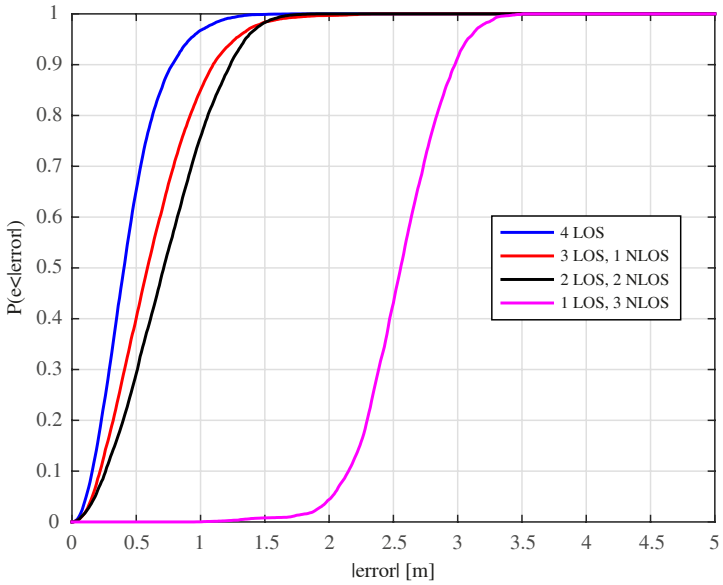


Figure 2.16: CDF of the position prediction error when we lose the LOS with 1, 2 or 3 anchors nodes.

Also note that only anchor nodes in LOS situation contribute to obtaining a good estimation, while on the other hand, the nodes in NLOS condition only contribute as a small amount of noise in the position estimation. Therefore, in order to correctly estimate a position at least three anchor node measurements in LOS situations are needed. However, our system achieves good results even when working with only 2 LOS anchors, since it avoids estimations outside the perimeter of the scenario. When we only have one anchor in LOS situation, the algorithm achieves noisy estimations, as shown in the last pink line in Figure 2.16.

Note that in the first and third experiments we took RSSI measurements for a long period of time, 10 000 s (and iterations), in order to obtain smooth CDF curves.

2.2.3 ZigBee Overview

ZigBee is wireless technology based on the IEEE 802.15.4 standard. It is mainly intended to create Wireless Personal Area Networks (WPANs) with small low-power digital radios.

Originally it was created as an alternative to the Bluetooth Classic or WiFi technologies during its first standardization between 1998 and 2003, appearing as a simpler and less expensive alternative. It was intended for applications such as wireless light switches, remote in-home devices, traffic management systems, and many other kind of equipment in both consumer and industrial sectors which require short-range and low-rate wireless data transfers.

ZigBee has a defined rate of up to 250 kbps, perfectly suited for sporadic data transmissions from a sensor network or from wireless peripherals. There are three possible ISM bands allowed in its PHY layer, the 2.4 GHz (worldwide) band supporting 250 kbps per channel, the 915 MHz (American and Australian) band allowing 40 kbps per channel and, finally, the 868 MHz (European) band with 20 kbps per channel. In these different bands there are also

differences in the number of available radio channels and channel bandwidth (up to 16 channels 2 MHz wide in the 2.4 GHz).

It can transmit typically from up to $10 - 100$ meters in LOS condition, depending on the transmit power and scenario characteristics, and for covering longer distances ZigBee allows data to be passed through a mesh-network (of intermediate nodes) to reach more distant devices. These and other functionalities were added to the upper layers of the ZigBee stack during its different version standardizations, the first in 2003, then a revised version in 2006, and finally the ZigBee Pro version in 2007 (fully backward-compatible with ZigBee 2006 devices).

The ZigBee SIG also defines three types of device:

- ZigBee Coordinator (ZC): It is the most capable device of the network in charge of coordinating the rest of the network as a root device. There is only one coordinator in each network since it is the initiator device, which also stores information about the network (including a *Trust Center* and a repository of security keys).

- ZigBee Router (ZR): This node is always on and listening to the network, as it usually acts as an intermediate router, passing on data from other devices.

- ZigBee End Device (ZED): Contains a minimum of functionality to communicate with its parent node (either the *coordinator* or a *router*). It is usually in low-power mode to save battery life so it cannot relay data from other devices. The ZEDs require the least amount of memory and hardware resources so they can be implemented with a lower-cost System-on-a-Chip (SOC) than the ones used to implement *routers* or *coordinators*. They usually implement battery-powered devices, while *routers* and *coordinators* are always mains-powered devices.

For the interest of our research work shown next in this chapter, we did not require the ZigBee upper layer mechanisms of mesh-networking or any other high-level routing capability. For ranging purposes we only require RSSI raw extraction from the ZigBee transceivers, used when implementing our architecture *Physical* layer. Therefore, we only used their two lower PHY and MAC layers in order to allow addressing and data transfer among the WSN nodes, to then extract the RSSI levels for our measurement campaigns and other experiments.

Integrating ZigBee Hardware with the RTLS Architecture

In the case of ZigBee technology, the WSN and its operation would be similar to Figure 2.17. Although there are several possible approaches when collecting measurements, a *coordinator* node [41] of the network usually acts as a gateway, forwarding every packet it receives directly to a host. Typically, RSS measurements are generated by mobile nodes when sending broadcast packets (as shown in step 1 in the figure). They are collected by the anchor network (and the

gateway) and transported wirelessly to the Host using the gateway node in a second step. Any individual anchor that receives a packet with the desired format can decode it and route/relay this RSS data to the Gateway (as shown in step 2 in the figure). It is necessary, therefore, to implement a small packet relay logic at the anchors and a ZigBee *port* at the Host, where the received RSS data, transported in the payload of the received packets would be parsed and sent to the *stdout* as: *Timestamp* (the current system time), *TransmitterID* (often a mobile node address), *ReceiverID* (anchor address) and *RSSI*.

In this particularly case, when using ZigBee technology to implement the *Physical* layer, we define three types of nodes:

- **Mobile node:** It broadcasts packets periodically with a certain random duration of some milliseconds added, in order to avoid collisions with other mobile nodes (maybe transmitting at the same scenario). And it enters sleep mode after sending each packet to ensure a higher autonomy of its batteries. Therefore, a sleep period actually marks the period of advertising plus some randomly time set between a minimum and a maximum value to avoid collisions with other mobile nodes.
- **Anchor node:** It has a fixed and a known position and it listens for packets sent by mobile and maybe other anchor nodes. This information coming from the mobile devices should be relayed by means of the logic explained later in Sections 2.2.4.2 and 2.2.4.2.
- **Gateway:** It is a special anchor node, which only listens for packets sent by the other nodes (anchors of mobiles), so it never relays any packets wirelessly. It is directly connected by cable to a Host, where the *Physical* layer of the proposed architecture would be attending to new RSS measurements, and possibly consumed by a *Location* client, remotely.

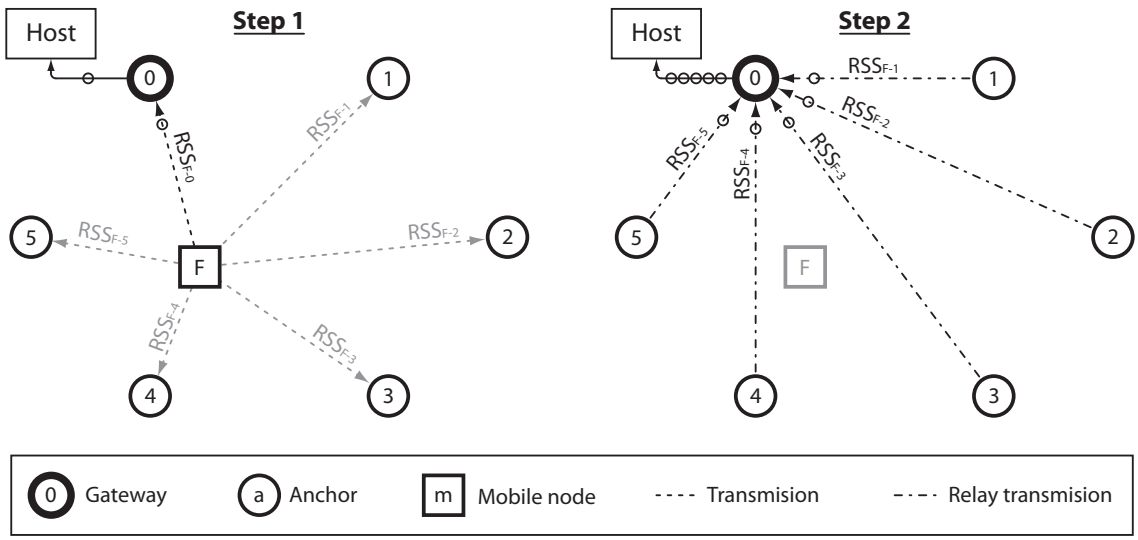


Figure 2.17: Process for getting RSS measurements using a ZigBee sensor network.

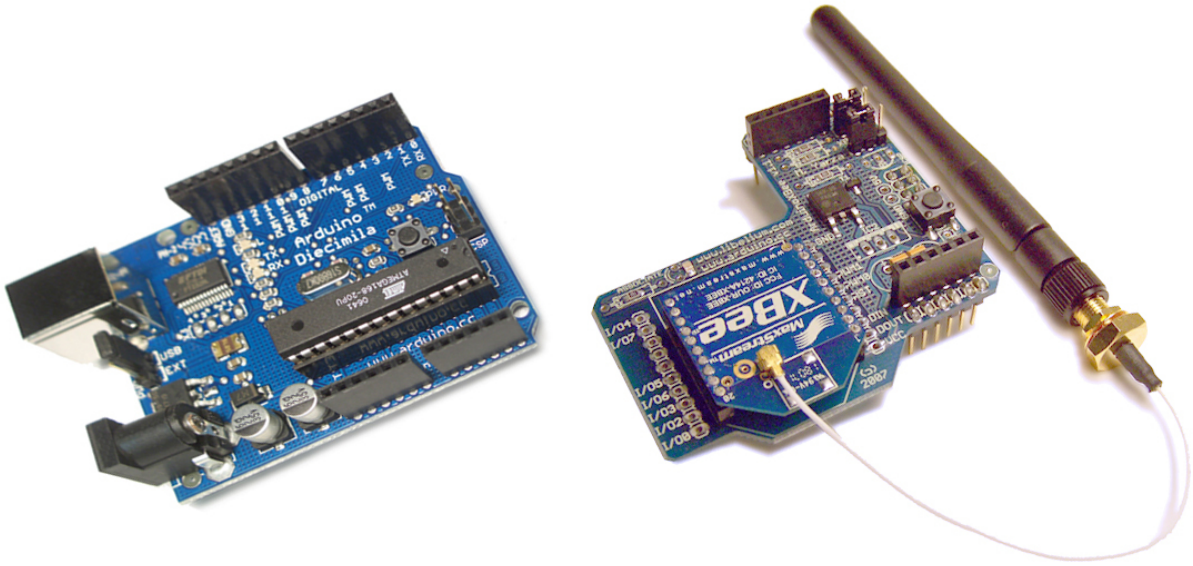


Figure 2.18: Arduino board, XBee on a XBee-Shield and 2.4GHz antenna (from left to right).

To implement the WSN we have chosen the combination of Arduino [43] boards (which contain an 8-bit Atmel microcontroller) and XBee Series 1 modules [37], mainly due to their low cost. Particularly, the XBee Series 1 module version used does not implement all the ZigBee layers, only the bottom layers, MAC and PHY following the IEEE 802.15.4 standard. However, these PHY and MAC layers are enough for the low level wireless communications that we need for ranging purposes. No other mesh and multi-hop techniques supported by the ZigBee standard are needed or used in our implementation. A photo of these two parts is shown in Figure 2.18.

All network nodes use the same hardware, which simplifies implementation and deployment. The only difference among nodes in the WSN are the firmware flashed into the Atmel microcontroller of each node, except for the Gateway node that does not need any microcontroller because it is directly connected to the Host by cable. The Gateway redirects all incoming data (wirelessly received) to a Host via USB port, in this case by taking advantage of the FTDI FT232R [46] UART-to-USB converter available in the Arduino board.

Figures 2.19, 2.20 and 2.21 show the interconnection schemes for the three types of nodes: mobile node, anchor and gateway.

Both the gateway and anchor nodes must be mains powered as they are always active waiting for new packets and they never enter a low power mode (sleep).

The mobile nodes, however, have their XBee modules configured to enter a cyclic sleep mode by means of a sleep period marked by the Atmel micro-controller (when the hardware is first initialized after every power cycle). Moreover, its Atmel micro-controller also enters deep-sleep mode when it finishes sending this initial configuration data to the XBee module via UART. Therefore, in this case a wire connection between the XBee module $ON/\overline{SLEEP}$ pin and the micro-controller $INT0$ pin must be added, so that the XBee can wakeup the

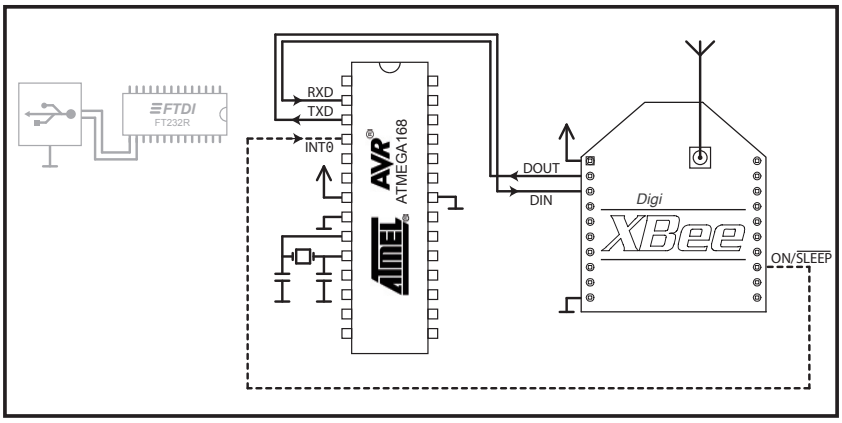


Figure 2.19: Mobile node hardware connections.

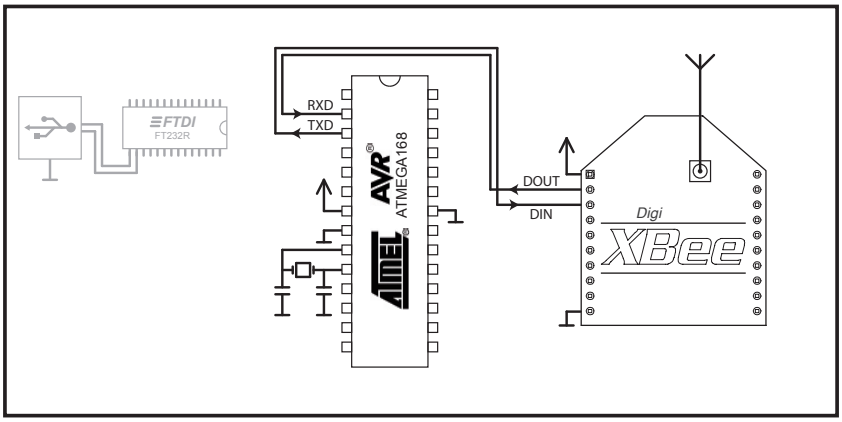


Figure 2.20: Anchor node hardware connections.

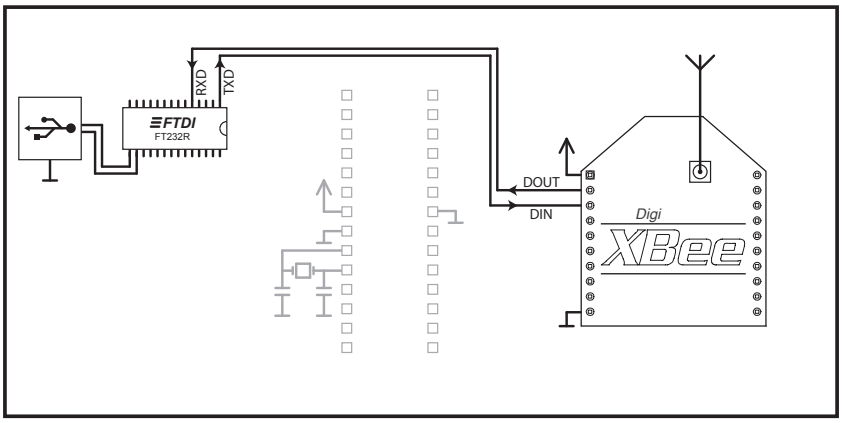


Figure 2.21: Gateway hardware connections, where the XBee module is connected to the Host by USB port, taking advantage of the FTDI UART-USB IC converter provided by the Arduino board.

microcontroller when its cyclic sleep period expires. This would provide a higher autonomy to the mobile nodes when they were battery powered, but its power consumption will be less than optimum as the Arduino Diecimila DC-DC linear regulator is not prepared for a really ultra low-power applications. It has a high quiescent current consumption of several milliamps, instead of the few microamps power consumption typical of a DC-DC switched regulator. Note that at the time of performing these experiments no newer Arduino boards with switched power supplies were available, but nowadays they are readily available.

Finally, the gateway node works without the help of the Atmel micro-controller available in the Arduino board, as shown in Figure 2.21. And all data wirelessly received by this node is passed to the Host via USB port, in near real time.

2.2.4 ZigBee Implementations and Experiments

2.2.4.1 Application to Bayesian Filtering Methods for Target Tracking in Mixed Indoor/Outdoor Environments

We apply the proposed architecture for the experimental evaluation of Bayesian filtering methods for target tracking in mixed indoor/outdoor environments [65], using a combination of ZigBee and/or GPS technologies. In this work we mainly contributed with the experimental setup configuration and the ZigBee and GPS hardware integration with the architecture for obtaining the different measuring campaigns at static positions (used for fitting model parameters) and performing several experimental trajectories for a moving target.

We tackle the design of a tracking algorithm that can work both indoors and outdoors, using GPS and/or RSS data collected from a ZigBee WSN. An SMC methodology (also known as particle filtering) is used as a general framework, which enables us to deal with a variety of different models (representing various indoor and outdoor scenarios) both for the observations and for the target motion, possibly switching between them using the general scheme of [63]. However, we also exploit the availability of GPS data and the ability to process them using Kalman filtering in order to (a) simplify the complexity of the tracker (when only the GPS technology is available) and (b) design efficient particle filtering algorithms for the online fusion of GPS and RSS observations. The superior performance of the resulting methods, when compared to outdoor GPS-only trackers, is demonstrated using experimental data. Moreover, synthetic observations are also generated in order to study, by way of simulations, the performance in mixed indoor/outdoor environments.

For more details about the implemented models and methods please see [65].

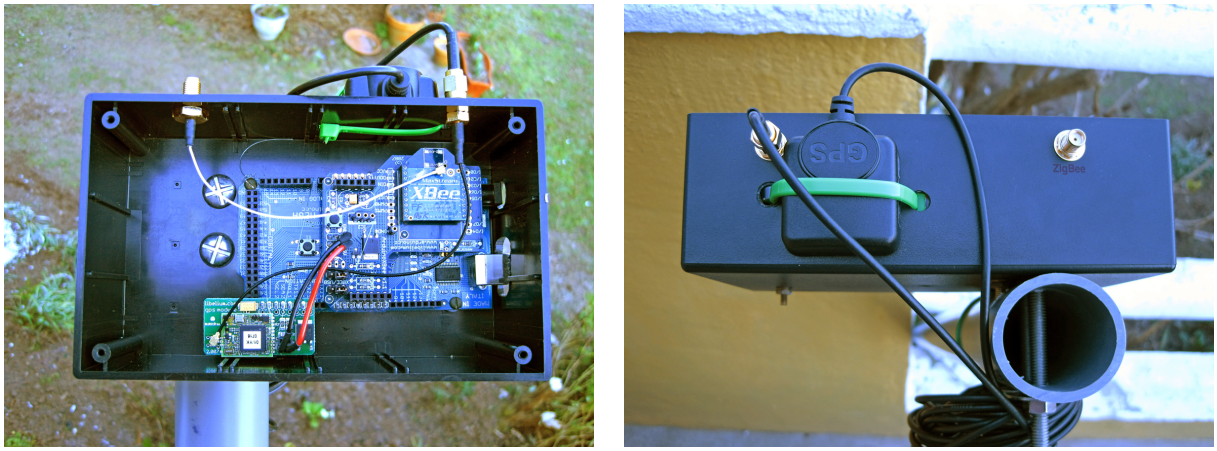


Figure 2.22: Mobile node made with an Arduino Mega, XBee and GPS receivers, mounted on a pole.

Experimental Setup and Results

We have developed two kind of hardware nodes using standard parts. For the collection of RSS data, we set up $J = 8$ anchors, acting as ZigBee transmitters, built using Arduino boards and XBee (series 1) modules (IEEE 802.15.4 compliant). The mobile device, which acts as a target, uses an Arduino Mega board and an XBee (Series 1) module, with an additional GPS receiver. All the 2.4 GHz band antennas were equal, 2 dBi monopole omnidirectional antennas, vertically oriented. For the GPS receiver we used an Right Hand Circularly Polarized (RHCP) external antenna. Each node was mounted inside a plastic enclosure and attached to a 2 m long non-metal pole. They were used in order to minimize the interferences caused by the person carrying the mobile node. Figure 2.22 shows the mobile device set-up. The anchors look very similar, with a smaller Arduino inside and without the GPS receiver attached.

Both the ZigBee and GPS measurements were taken in an outdoor scenario in the middle of

Figure 2.23: Outdoor scenario set-up picture.

a $600\,m^2$ clear field area (without walls or trees), close to the village of Cambre (in the province of A Coruña, Spain). We deployed an 8 ZigBee anchor network covering a 6×10 meter area and, on the other hand, one mobile node, with a ZigBee and a GPS receiver. This mobile node acted as a gateway, receiving both the ZigBee packets (from the 8 anchors transmitting each $150\,ms \pm 10\,ms$) and the GPS signal at $1\,Hz$ rate. The measurements were tagged by this node and sent to a small laptop by USB port, where a specific ZigBee *Physical* layer (and *port* application) were running. In this case, the gathered measurements were forwarded to different log files for future off-line processing, instead of being accessed in real-time from a remote *Location* client (see Section 2.1.2 for more details about the architecture).

The GPS measurements were sent directly to the Host capturing the GGA National Marine Electronics Association (NMEA) frame each second to the corresponding log files, correctly time tagged in between the ZigBee RSS measurements.

A picture of the scenario set-up is shown in Figure 2.23 and a logical schema is shown in Figures 2.24 and 2.25, following the connection schema and nomenclature shown in previous Section 2.2.3 for this particular scenario.

In order to test the performance of the proposed algorithms and to be able to build realistic observation models we collected numerous RSS and GPS observations in the above mentioned

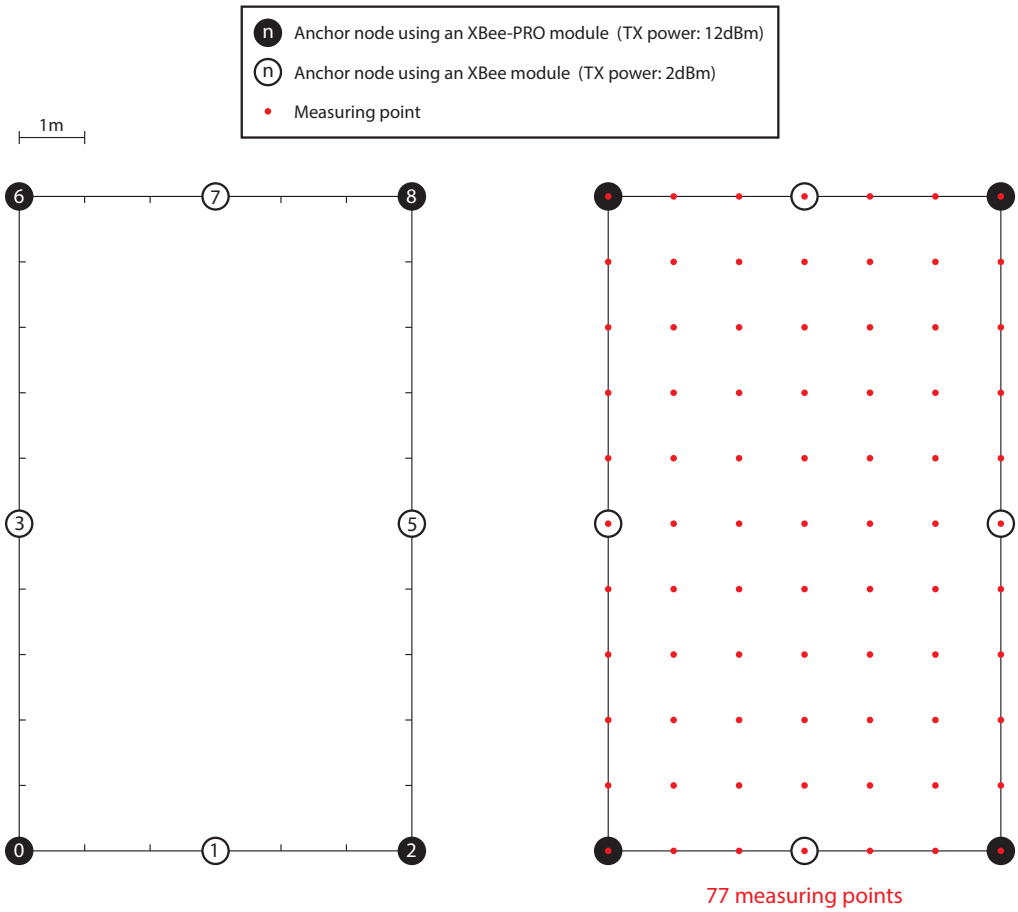


Figure 2.24: Scenario set-up schema.

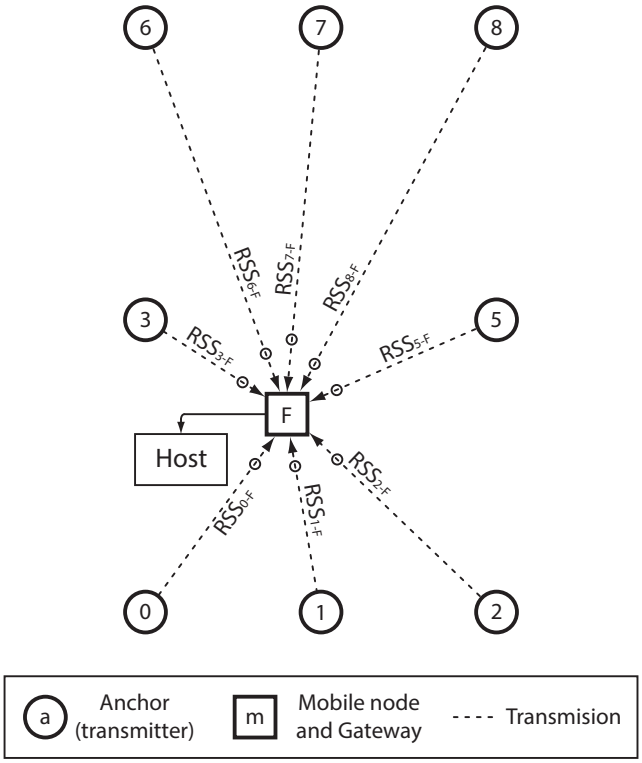


Figure 2.25: Scenario connection schema.

area. Figure 2.24 shows the 77 positions where both ZigBee and GPS measurements were collected regarding the 8 anchor nodes. Note that we used 2 types of XBee Series 1 nodes (XBee and XBee-PRO), the only difference of being that they are able to transmit at different output power. In Figure 2.24 we represent the XBee-PRO (transmitting at 12 dBm) in black, and the normal XBee ones (transmitting at 2 dBm) in white.

Figure 2.26 shows experimental data taken from sensors 1 and 8 and the log-distance path

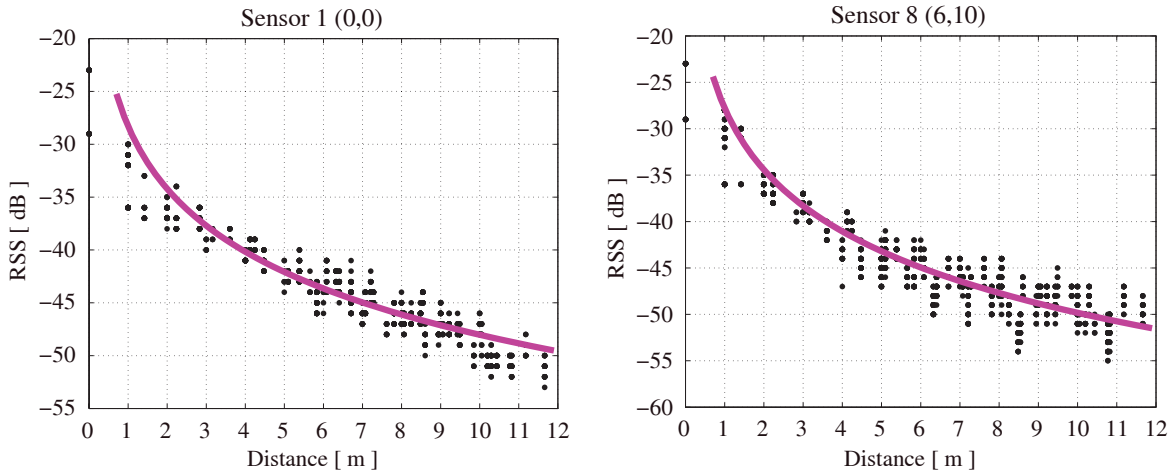


Figure 2.26: Real ZigBee outdoor measurements taken in an outdoor environment for Sensors 1 and 8 and the observation functions adjusted to them.

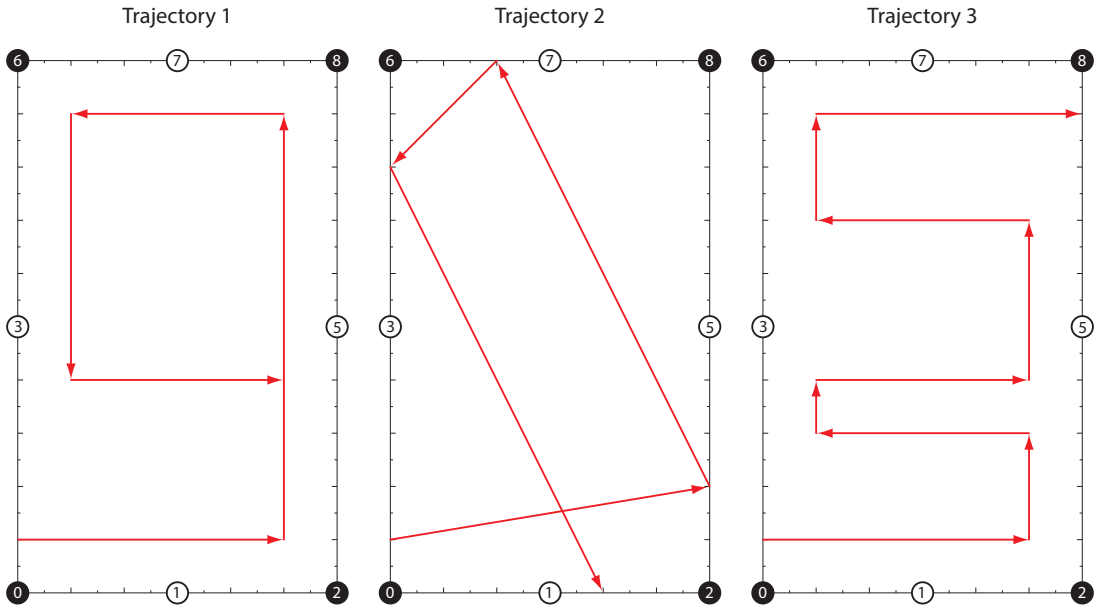


Figure 2.27: Experimental trajectories.

loss models adjusted to them. Note that we have fitted one model for each sensor. The modelling of the ZigBee observation functions to fit these real measurements can be found in [65].

In order to illustrate the performance of the outdoor algorithms we have also taken experimental data from the 8 ZigBee anchors and experimental GPS data from a moving target following three specific trajectories shown in Figure 2.27.

Figures 2.28, 2.29 and 2.30 show a tracking example of the two outdoor algorithms: The sub-figures in the left-hand side show the target trajectory estimated by the Kalman filter (when using only GPS observations), while the sub-figures on the right-hand side show the trajectories estimated by the SIS filter (when ZigBee RSS and GPS data were combined). The real trajectory is drawn with a dark coloured line, while the estimated trajectory is drawn in a light coloured line and, finally, the 8 ZigBee anchor nodes are depicted with square points.

As we can observe, the trajectories with GPS data have a greater error and we can obtain a high degree of improvement by incorporating RSS data. This is as expected, because conventional GPS receivers (non-differential and uncorrected using any reference base station) suffer from a typical inaccuracy of 5 to 10 meters depending on open sky conditions [93, 147]. The area in which we perform the tracking is, therefore, too small for the precision that the technology provides. The ZigBee technology instead has a much lower range but achieves a greater precision.

In order to illustrate the performance of the tracking scheme we have generated synthetic trajectories that switch between environments (from indoor to outdoor) and we have synthetically generated observations that switch between technologies (from indoor ZigBee to outdoor ZigBee plus GPS, to outdoor with GPS only).

We have simulated a scenario consisting of an indoor room of dimension 6×10 meters,

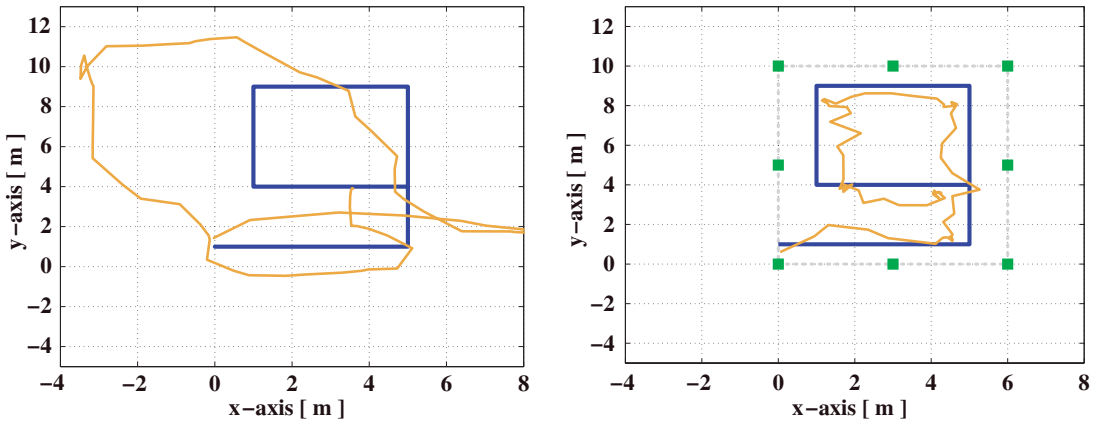


Figure 2.28: Outdoor tracking example using ZigBee and GPS real data. The figure on the left-hand side uses GPS only for the estimation (via the Kalman filter) and the figure on the right-hand side uses both GPS and ZigBee data (via the SIS algorithm).

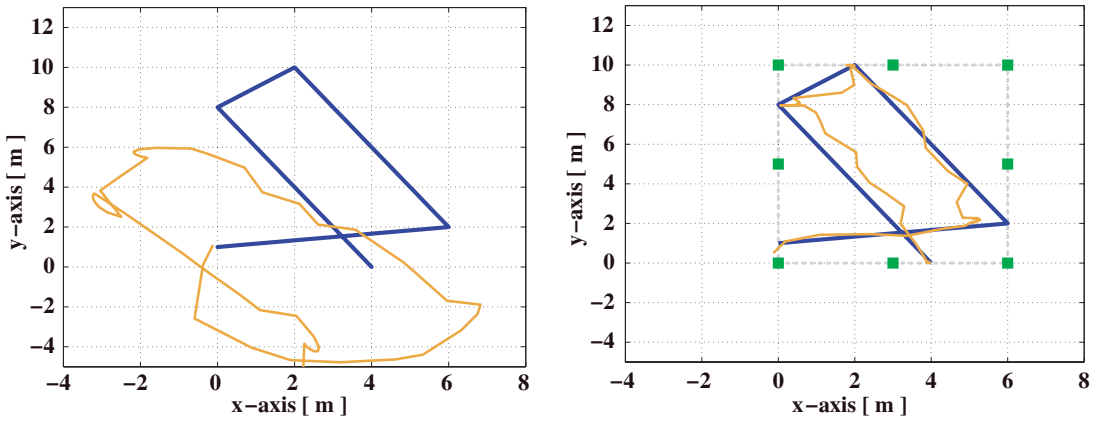


Figure 2.29: Outdoor tracking example using ZigBee and GPS real data. The figure on the left-hand side uses GPS only for the estimation (via the Kalman filter) and the figure on the right-hand side uses both GPS and ZigBee data (via the SIS algorithm).

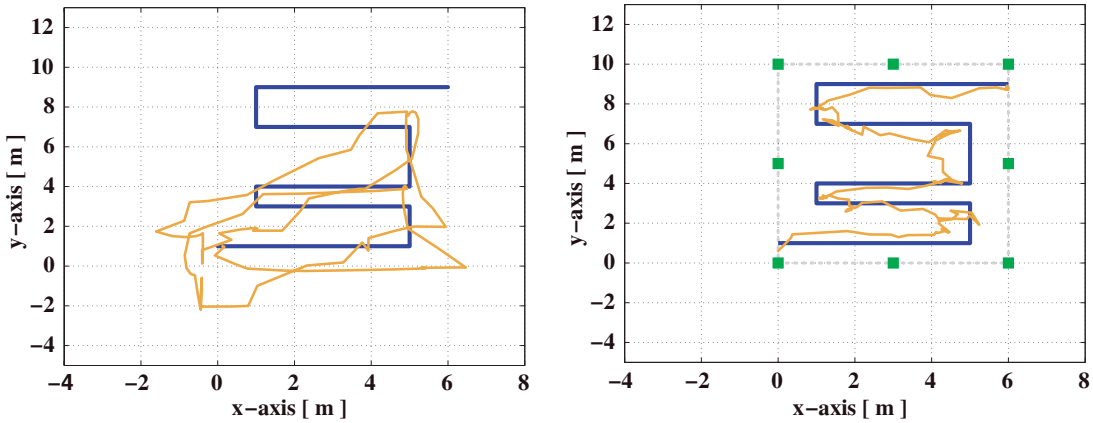


Figure 2.30: Outdoor tracking example using ZigBee and GPS real data. The figure on the left-hand side uses GPS only for the estimation and the figure on the right-hand side uses both GPS and ZigBee real data.

linked to a 2 meter long and 2 meters wide corridor that leads to an outdoor environment, which is also an area of 6×10 meters. This is shown in Figure 2.31. In the 6×10 meters area on the right and in the corridor we assume we are in an indoor environment. In the 6×10 meter area on the left we assume we are in an outdoor environment.

A random trajectory that switches between environments, generated according to the corresponding dynamic models, has been randomly generated for exemplification. At each time instant t, we check the position of the target in the previous time instant, $\mathbf{r}_{t-1}$, and the new position, $\mathbf{r}_t$, is generated according to the environment-specific dynamic model. In order to generate the observations that switch between technologies we also check where the target is at every time instant, and we generate the observations according to the environment-specific model. On the other hand, as we have two observation models for the outdoor environment (GPS only and GPS plus ZigBee), we have introduced a new random variable which in outdoor environments chooses the available technology. When $K_t = 1$ we assume we only have GPS technology available and when $K_t = 2$ we assume we have the two technologies and, therefore, simulate both types of observations. We have also associated a transition probability for the technology selection variable so that we do not change technologies too fast, that is, $p(K_t = i|K_{t-1} = i) = 0.95, i = 1, 2$.

Figure 2.31 shows a tracking example in this mixed scenario. The dark line is the true trajectory, while the light line is the estimated one, and the square dots represent the ZigBee anchors. The beginning of the simulation is situated in the right-side *indoor* room, and the simulation lasts for 300 seconds. The observation sampling period is fixed to $T = 0.4$ seconds. In the first 200 seconds the target moves in an indoor environment when we use the indoor specific algorithm. From the 200-th second, the target moves though the corridor and then it goes out to the purely outdoor environment. In the corridor we assume we have measurements from both the GPS and the outdoor ZigBee WSN available. In the left-side outdoor environment we have allowed variable K_t to choose which technology is available at each time instant.

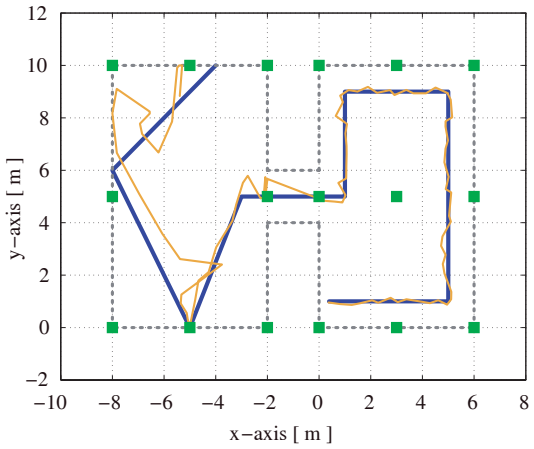


Figure 2.31: Example of tracking a simulated trajectory using different technologies.

2.2.4.2 Cross-Measurement Process to Increase the Available RSS Measurements

The performance of any location systems based on WSN using RSS information crucially depends on the quality of the measurements obtained and how received signal levels adapt to a particular propagation model [127]. It is well known that these power levels randomly vary according to environmental conditions (obstacles, moving people, loss of the line-of-sight, etc.).

When using a propagation model, even when it sufficiently fits the environmental propagation conditions, there are other factors, such as the loss of line-of-sight among nodes, which produce significant changes in the propagation conditions [130]. These factors can make the multipath components acquire great importance and, therefore, cause the considered models to lose their validity.

Some location systems are typically based on the periodic transmission of pilot packets from mobile nodes, which are captured by an anchor node network to extract RSS information. Typically, only signal levels between each individual mobile node and the anchors are used for localization, because they are usually enough to estimate the mobile node position by means of a more or less advanced location technique (multi-lateration [75], particle filtering [91][68], etc.). And all of these techniques always suffer some different problems when the measurements do not adapt to the considered propagation model.

As a possible improvement to the propagation issues faced by these location systems due to environmental changes, we proposed a measuring process to increment the available number of RSS measurements, in order to better detect these environmental changes. We can do so by also detecting the ranging conditions among the anchor network, and when we detect any important changes among them, we can make all kind of decisions, as for instance ranging corrections with respect to a certain node, or performing a location algorithm dynamic calibration giving more or less weight to some of the anchors, etc.

In this section we introduce an easy and economical way to build a ZigBee based WSN for obtaining cross measurements among all network nodes, not only among mobile and anchor nodes, with the following objectives:

- Efficient and simple firmware code, with low computational cost and CPU time, that could be added to any already existing firmware implementation in the nodes.
- A cheap and small-sized hardware. In our case we mounted an XBee on an Arduino.
- A high immunity to collisions. We send packets periodically from the mobile nodes but with some period random variation of several milliseconds to avoid packet collisions.
- The ability to increase the number of anchor nodes on the fly and thus improve system accuracy without reprogramming mobile or anchor nodes, or relaunching the location application.

- The mobile nodes should have a very low power consumption, to be battery powered.

- The output format of the obtained measurements should be compatible with the ZigBee *Physical* layer of the proposed architecture (see Section 2.1.2 for further details).

This section is divided into the following subsections. First, we explain in depth the packet relay process among the different nodes of the system. Next, we show how the packet relay algorithm is implemented in the anchors. Finally, we explain how the host processes the received frames via UART and how it creates the RSS table with the cross measurements among all the WSN nodes.

Packet Relay Process

The raised cross measuring process is divided into three steps whose operation is independent of the number of anchors in the WSN. Figure 2.32 illustrates these three steps when using a 4 anchor WSN. For simplicity, the raised process is presented in an ideal situation when all the nodes are in range and there are no packet losses due to interferences with other wireless networks. However, this explanation can be extrapolated to larger networks, where only RSS data of closest nodes (in range) is collected.

As explained in more detail in Step 2 in this section, the notation used in the three steps to represent the *Payload* data of each transmitted packet is the following: `Source_Addr(Payload_RX)Current_RSSI`. So, the *Payload* is the part of each packet which actually transports the ranging information.

Step 1

A mobile node that is in a sleep state automatically wakes up when its XBee cyclic sleep period expires. First, the XBee wakes up the Atmel micro-controller (also in sleep state) by posting a hardware interruption (using its $INT0$ interrupt line). This is automatically performed by the XBee firmware rising its $ON/\overline{SLEEP}$ output line after it wake up.

Then, the Atmel microcontroller resumes its execution at the same point where it left before going to sleep and **broadcasts** a small packet to the network (by using a destination address of `0xFFFF`). This packet just has 1 irrelevant byte in its *Payload* and will be received by all nodes within range in the network (assuming no collisions with other mobile nodes).

Afterwards, it calculates the new pseudo-random sleep period for the XBee module, between predetermined minimum and maximum values fixed in the code. It passes this to the XBee module by sending an `ATSP` (Sleep Period) command [38] (with possible values between 10 ms and 268 s).

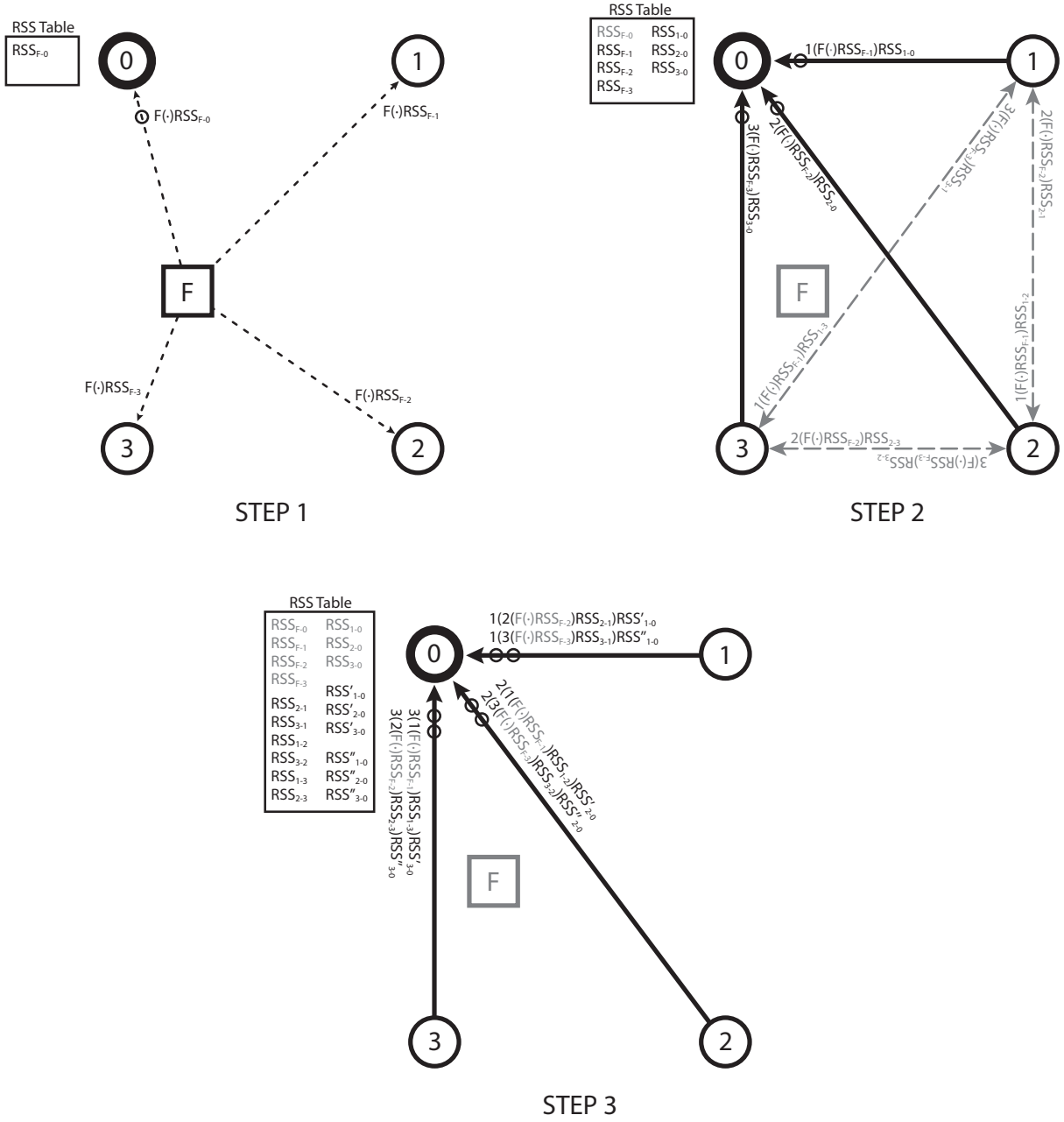


Figure 2.32: Cross-measurement process divided into three steps.

And finally the Atmel itself switches off all the unneeded hardware parts (ADCs, I2C and SPI ports) and enters into *Power-down* mode [1] again, the lowest sleep state available, with a current consumption of less than 5 μA.

When this Step 1 is completed, each anchor node close to the mobile node should have detected the same receive packet (RX) but with a different RSS level. Note that if the Gateway (node with address 0) and the mobile nodes are in range, we will get the RSS information about the link F→0, denoted by RSS_{F-0} (between the transmitter F and the receiver 0).

Figure 2.33 shows the structure of each *RX* (Receive, `API ID.=0x81`) frame particular to

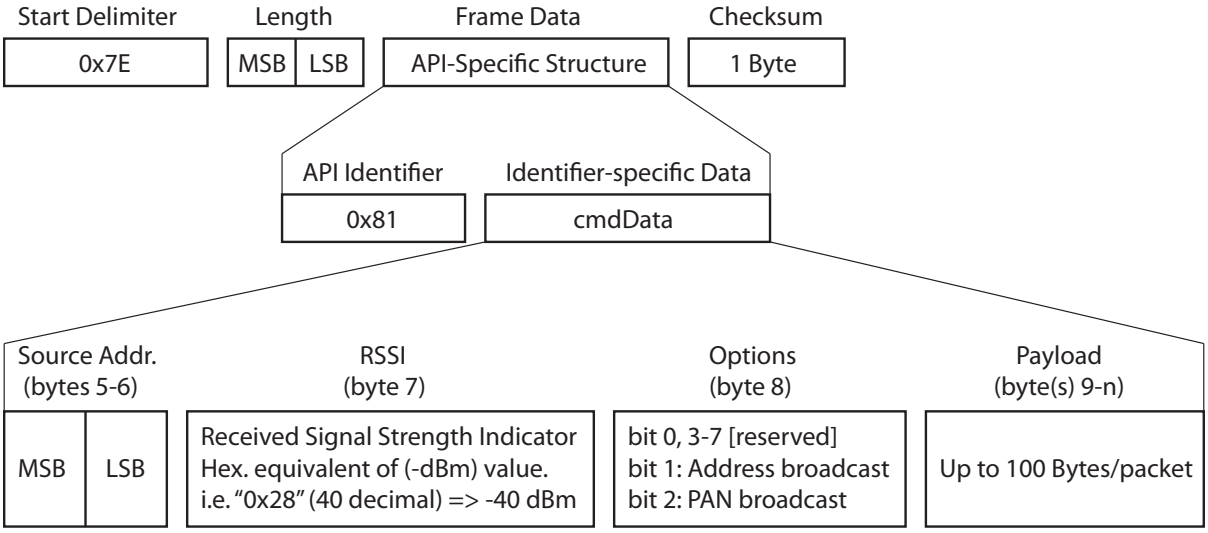


Figure 2.33: RX (Receive) packet: 16-bit Address.

the XBee nodes, where the byte 7 contains the `RSSI` level (in dBm) and bytes 5 and 6 contain the `Source Address`, which identifies the broadcaster transmitting node. The `Length` field includes the total length of `Frame Data` so that we can easily calculate the length of the *Payload* by computing $\texttt{Length} - 5$. That is, the `Frame Data` length minus 1 byte of `API ID.`, 2 bytes of `Source Addr.`, 1 byte of `RSSI` and 1 byte of `Options`. So, in the `Payload` (of up to 100 bytes/packet) we can accumulate different RSS information after each process iteration, and decode it back in the *Physical* layer of our architecture. As always, this is implemented by using a *port* program available at the Host computer, responsible for attending the data coming from the *Gateway* node.

Step 2

Any anchor node in range with the mobile node receives a packet from the previous step. It then detects that the received packet is of the right length to be relayed, and finally it creates a new packet by accumulating this information into a new packet *Payload* in the following manner:

$$\texttt{Payload\_TX} = \boxed{\texttt{Source\_Addr(Payload\_RX)Current\_RSSI}}$$

For simplicity, we will use the `Payload_RX` notation to refer to the Payload of the received packet (from the previous step) and `Payload_TX` for the new expanded Payload. Note that the length of a new *Payload* is always equal to the size of the previous $\texttt{Payload\_RX} + 5$, because we add 2 bytes of `Source_Addr`, another 2 characters '`(`' and '`)`' for readability purposes, and 1 byte of the current `RSSI` to be relayed.

The anchors then wait for a pseudo-random period of time (between a minimum and a maximum default value) and simply re-broadcast the new composed packet. This pseudo-random time value is also set to avoid collisions with other anchors.

All nodes in the network will receive $N_a - 1$ packets after this step, with N_a the number of anchor nodes (not including the Gateway node).

After this second step is completed, the Gateway has RSS information of the links between the mobile node (F) and all the anchor nodes in range. In addition, it also has cross information of the anchors that are 1-hop from it (RSS_{1-0}, RSS_{2-0} and RSS_{3-0}), but still does not have any information of the links between the other anchors (links RSS_{1-3}, RSS_{1-2} and RSS_{2-3}).

Step 3

In this step, all the anchor nodes first detect that the received packet in the previous step is of the right length to be relayed, and if not it would be discarded. They create a new packet the same way as in the previous Step 2, by putting the `Source Addr.` in front of the previous `Payload_RX` and the new detected `RSSI` level. They wait another pseudo-random period and then transmit the new packet. In this case, instead of broadcasting it, they **unicast** it directly to the Gateway node (by simply sending it to the destination address `0x0000`), which is always the Gateway node, to avoid the other anchors repeating the detection process, with an incorrect length, needlessly.

Upon the completion of this step, the Gateway has the RSS information relayed among all the nodes (mobile and anchors) in the WSN. It would have this information duplicated for all the links at more than 1-hop from it (RSS_{x-y} and RSS_{y-x} links), and N_a times replicated for the direct links at 1-hop from it (RSS_{x-0}). Note that each intermediate relay brings a new RSS

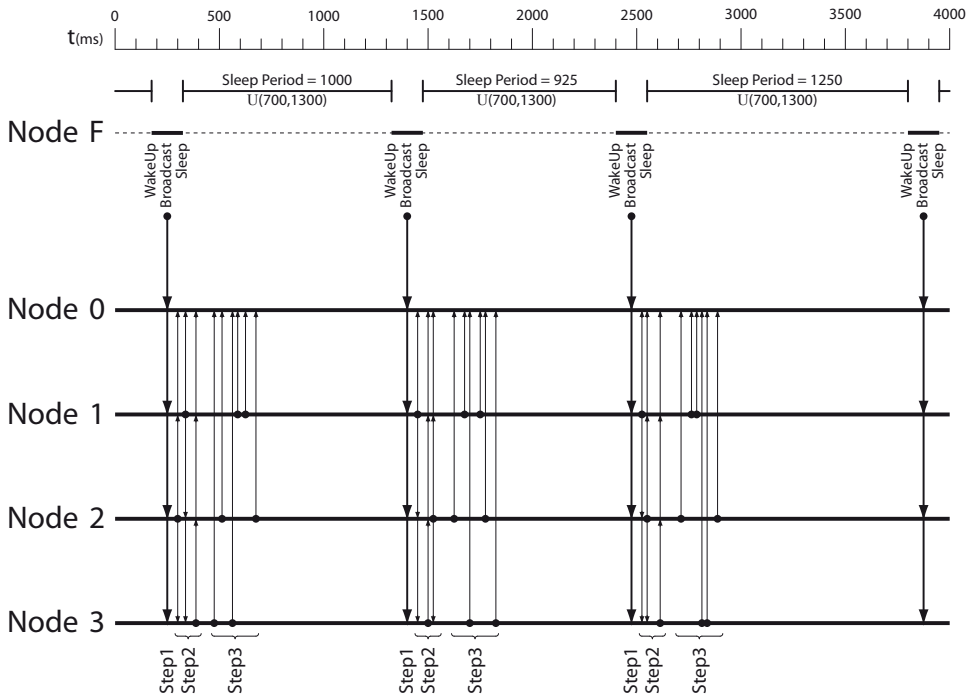


Figure 2.34: Runtime diagram of the cross measuring process.

level, as there is 1 relay in Step 2 and $N_a - 1$ relays in Step 3.

When the mobile node re-awakens after the XBee cyclic period is reached, all this process starts again, as shown in Figure 2.34.

Anchor Packet Relay Algorithm

The pseudocode in Figure 2.35 represents the overall structure of the code stored in the anchor nodes, which act as packet relayers. The code is contained within an infinite *loop()* function, divided into 2 main parts which are processed each time some bytes appear in the UART RXD pin of the Atmel microcontroller [1], coming from the XBee module (DOUT pin):

- **Packet detection:** We read the header data for incoming *RX* received packets (with `API ID.=0x81`). If something different is detected then we simply clean the UART input buffer and wait for new packet bytes to arrive.

```
void loop() {
  if (Serial.available()) {
    ////////////////////////////////////////////////
    //              Packet detection              //
    ////////////////////////////////////////////////
    Read one byte
    if (the first byte != Start_Delimiter 0x7E) {
      // Wrong start delimiter => Empty serial buffer
    } else {
      Read Frame-Data Length (bytes 2-3)
      Read API Identifier (byte 4)
      if (API Identifier != RX_Packet 0x81) {
        // Wrong API Id. => Empty serial buffer
      } else {
        Read Source_Addr (bytes 5-6)
        Read RSSI (byte 7)
        Read Options (byte 8)

        for (i=0; i<Length-5; i++) {
          Read Payload_RX[i]
        }
        Read Checksum (1 byte)

        ////////////////////////////////////////////
        //          Packet relay code             //
        ////////////////////////////////////////////
      }
} } }
```

Figure 2.35: Packet detection code

- **Packet relay:** We detect if the received packet must be relayed. According to the `Length` of the packet, we know if we are in *Step 2* or *3* and, therefore, if we have to send the new packet by broadcast or unicast (to the node Gateway). The `Payload_TX`

```
//////////////////////////////////////////////////
//                Packet relay code                //
//////////////////////////////////////////////////
//
// The possible packet lengths are:
// 6  = 5 + [1 byte in Payload]       <-- in Step 2
// 11 = 5 + [3+(1)+2 byte in Payload] <-- in Step 3
//
// We compose the new Payload as:      Source_Addr + '(' + Payload_RX + ')' + RSSI
//                                          2          1     Length-5     1      1
// New payload length: 2+1+(Length-5)+1+1 = (Length)
// New packet length: (Length)+5
Length_TX = Length+5;

// We compose the new Payload_TX as:   Source_Addr + '(' + Payload_RX + ')' + RSSI
Payload_TX[0] = (byte)(Source_Addr>>8);
Payload_TX[1] = (byte)Source_Addr;
Payload_TX[2] = '(';

for (i=0; i<Length-5; i++) {
  Payload_TX[3+i] = Payload_RX[i];
}
Payload_TX[3+i] = ')';
Payload_TX[4+i] = RSSI;

// We calculate the new Checksum, as XBee defined: "Not including frame delimiters and length,
// add all bytes keeping only the lowest byte of the result, and substract from 0xFF".
//
// The new TX packet has this structure:  API_ID=1, Frame_ID=0, Destination_Addr,
//                                        Options=0 and Payload
// Checksum = FF - (1+0+Destination_Addr+0+Payload) & FF
if (Length==6) {   // We are in "Step 2" -> Send Broadcast relay
  Destination_Addr = 0xFFFF;
  accu = 0xFF; //byte accu = 0x01 + 0xFF + 0xFF
} else {           // Else, we are in "Step 3" -> Send relay to Gateway
  Destination_Addr = 0x0000;
  accu = 0x01; //byte accu = 0x01 + 0x00 + 0x00
}

for (i=0; i<Length; i++) {
  accu += Payload_TX[i];
}
Checksum_TX = 0xFF - accu;

delay(random(30,50));  // We wait for a small random delay, to avoid collisions
send_packet_TX();      // We relay the new generated packet
```

Figure 2.36: Packet relay code

is composed (with the format explained in *Step 2*) and with a new `Checksum`. Then we send the new *TX* packet (using the `API ID.=0x01`) [38]. This is achieved by just writing the new packet bytes out to the Atmel UART with an extra *Carriage Return* (0x13) byte as an end of packet mark. These bytes will go from the Atmel *TXD* pin to the XBee *DIN* pin.

Frame Processing in the Host

The Gateway node is connected to the Host by USB cable as explained in Section 2.2.3, taking advantage of the FTDI UART USB IC [46] available in the Arduino board.

The XBee module of the Gateway must be configured in *API* mode [38] with an UART speed configured high enough to be able to process several frames per second, maybe coming from multiple mobile nodes. We decided to use a 115200 bps port speed.

All packets received wirelessly go out through the *DOUT* pin of the XBee module (in the Gateway) and pass to the Host through the *FT232R* converter.

The ZigBee *port* available at the Host finally parses the incoming data with the common format explained in Section 2.2.3. In order to create an RSS table with the cross measurements among all network nodes, the goal is to print this table to the *standard output* (stdout), according to the following format:

`Timestamp TransmitterID ReceiverID` $\mathbf{RSS_{TxID-RxID}}$

The extended ZigBee *port* implementation performs like this. First, from the Host we need to open a connection at 115200 bps speed with the according serial port of the UART-to-USB converter, and enter an infinite loop performing the following steps:

1. We decode *RX* frames (with `API ID.=0x81`) as explained in the pseudocode previously in this section, extracting the Frame-Data `Length` (bytes 2 and 3), `Source_Addr` address (bytes 5-6), `RSSI` level (byte 7) and `Payload` data (`Length` $-\,5$ bytes).

2. We get the current system time epoch (in miliseconds) in UNIX timestamp format [57].

3. We write these parameters to *stdout* with the **`Timestamp Source_Addr 0 RSSI`** format, where the *Source_Addr* at this step corresponds to the *transmitter node* address, and the **`0`** stands for the *Gateway receiver node* address.

4. We switch our code execution according to the *Length* of the received packet, which can be 6, 11 or 16 bytes long, as explained in the previous section, to detect when we are in Step 1, 2 or 3, respectively.

(a) **In case Length = 6:** We do not have to decode anything since the *Payload* is one byte long (containing irrelevant data) as the packet was directly sent from a *Mobile* node.

(b) **In case Length = 11:**

i. We extract the first 2 bytes of the *Payload_RX* that corresponds to *TransmitterID* of the previous hop.

ii. We extract the last byte of the *Payload_RX* that corresponds to the observed RSSI measurement, of the previous hop.

iii. We write these parameters to *stdout* with the following formatting: `Timestamp TransmitterID Source_Addr RSSI`, where in this case, the `Source_Addr` corresponds to the `ReceiverID` of the previous hop.

iv. We do not have to decode anything more because the remaining *Payload_RX* data is just one irrelevant byte.

(c) **In case Length = 16:** We follow the same process as in point b) until the remaining *Payload* length is 1. We unpack the *Payload* of each packet by removing the first 3 and last 2 bytes of the previous *Payload* and assign at each step *Source_Addr* $\leftarrow$ *TransmitterID* of the previous step.

2.3 Conclusions

In this chapter we introduced a distributed multi-layer architecture for implementing RTLS systems which can abstract the hardware technology accessed by different location algorithms, running maybe in different machines. The raised architecture is divided into four layers: A *Physical* layer accesses the WSN at low-level and abstracts the WSN hardware to other layers. A *Location* layer implements a location algorithm, and finally, another two layers (Proxy and Wrapper) deal with the remote calls and communications between the previous two layers (in a completely transparent manner), by using the XML-RPC standard. In this chapter we also present a Matlab client code showing how to transparently and remotely access the *Physical* layer, to obtain experimental RSS data.

The main benefits achieved are the re-usability and independence of the implemented layers, which can use different programming languages to be implemented, and the possible remote access to different WSN hardware from multiple *Location* clients. Therefore, we obtain a hardware-independent and distributed multi-layer software architecture for RTLS implementation.

We also show two possible ways of implementing a physical layer based on Bluetooth Classic and ZigBee wireless technologies, as well as different implementations and experiment for both technologies:

1. For the Bluetooth case we applied the proposed architecture for the experimental evaluation of a Bayesian filtering method, for jointly estimating the target position and the path-loss propagation model parameter. We introduced a mathematical state-space model which represents:

   - The dynamics of a target position and of the path-loss exponent, which can be switched between two states (LOS/NLOS).
   - The observation model which describes the relationship between the RSS observations and the target position.

   We recursively approximate the a posteriori pdf of the state using an SIR particle filter, and use an MMSE estimator plus a decisor for the path-loss exponent. This was chosen instead of a MAP estimator in order to obtain a good trade-off between computational complexity and estimation accuracy.

   By running different experiments we showed the effect of choosing a fixed and wrong path-loss exponent for an arbitrary anchor while the rest were correctly estimated, and how the PF could automatically adapt to a propagation change between the LOS and NLOS situations while the target position is being estimated.

2. For the ZigBee case we applied the proposed architecture for the experimental evaluation of Bayesian filtering methods, in this case for target tracking in mixed indoor/outdoor environments using the ZigBee and GPS technologies. In this work we mainly contributed with the experimental setup configuration and the ZigBee and GPS hardware integration with the architecture for obtaining the different measuring campaigns at static positions (used for fitting model parameters) and performing several experimental trajectories for a moving target.

   The experimental results present three different trajectory tracking examples for a moving target, showing the difference between only using GPS for the estimations, via a Kalman filter, versus the case where the ZigBee RSS observations are combined with the GPS positions, via a Sequential Importance Sampling (SIS) algorithm. Finally, some simulated results show a hybrid indoor/outdoor tracking experiment using the two different technologies when they are available at different time instants, so that they are used sometimes individually, and sometimes combined.

3. For the ZigBee case we also proposed a method for increasing the number of RSS available measurements, by obtaining cross RSS information among all the nodes in a ZigBee network.

   Typically, the main RSS information is given by the links among mobile nodes and the anchors, but with our method we can obtain ranging information among all the links

between the anchors. We can improve location algorithms performance by possibly using different dynamic recalibration techniques or propagation model parameter adaptation, taking into account this extra crosswise ranging information.

We show how this method can be implemented in only three steps using commonly used prototyping hardware such as an Arduino board and an XBee wireless module, with the advantage that all this logic and output frame format is fully compatible with the *Physical* layer formatting proposed in this chapter.

Finally, it is important to note that the proposed architecture can be improved in many senses since at this stage it only supports RTLS systems based on WSNs of fixed anchors locating mobile devices (not other schemas such as moving anchors or fingerprinting methods, for instance), and by only using RSS measurements (not ToA/TDoA, AoA or any other sensor-type information). It only supports a single technology at a time, it does not provide any high security mechanism for maintaining data privately, nor any persistence mechanism to store measurement data or estimated positions in a database (for later access).

In Chapter 3 we continue evolving this multi-layer architecture to a much higher level.

Chapter 3

Multi-Technology Location Systems

In this chapter we propose an approach to the problem of developing generic and easy-to-use hybrid Real-Time Location Systems (RTLSs) by introducing a distributed multi-layer and multi-technology software architecture. This chapter is based on the publication [131].

The proposed system is able to obtain measurements from several WSNs (RF nodes) and generic sensors (e.g. inertial sensors, motion detector, video cameras, etc.), obtaining position estimations by means of several location algorithms running simultaneously in real-time. It also provides a flexible data fusion module to furnish combined positioning information to several concurrent client applications. This fusion module can automatically merge positioning information from an arbitrary number of nodes (several WSN technologies) and sensors registered in the system. Moreover, communication between the clients and the server is made by using a well defined interface.

The rest of the chapter is organised as follows: Section 3.1 introduces an overview of the state of the art on solution to implement RTLS systems. Section 3.2 introduces the proposed architecture features. Section 3.3 presents a logic view of the elements that make up the architecture and how they are connected. Section 3.4 shows a runtime view of the architecture, presenting the common work-flow of tasks like the registration of WSN hardware, mobile nodes, sensors, etc., and the insert and query of measurements and position estimation information. In Section 3.5 we show two implementation example case studies; one based on two ZigBee and UWB networks, and an inertial sensor (accelerometer) to illustrate how to deploy a representative set of WSN technologies, nodes and sensors in a scenario; and another real scenario showing how to implement a fingerprinting solution based on WiFi technology. Section 3.6 shows some validation results using a simulated environment where we deploy two ZigBee and UWB WSNs, and how we track the position of a multi-technology target, comparing several data fusion methods. Finally, section 3.7 shows some conclusions.

3.1 Introduction

Typically, location-aware applications build the entire system (including sensing, representation, and application logic) as a monolithic structure. These monolithic systems are a fast-developing solution when considering a particular application and type of sensors, but they lack flexibility and scalability. Therefore, these solutions are difficult to generalize and retarget to new applications, increasing the costs of a further development. Authors in [107] were one of the first group who introduced the idea of a layered architecture for LBS. The seven layers try to provide a stack for any kind of location-based service or application.

In a more general way [139] described the LBS by means of a three layer model: positioning, middle-ware and application layer. The application layer determines the data usage for specific location applications, based on the data extracted from the middle-ware. The middle-ware provides the interfaces for the application layer, hiding technical details from the positioning layer. Finally, the positioning layer deals with the deployment, configuration and calibration of wireless location infrastructure; gathers raw data (radio signal strength, time of arrival, angle of arrival...); and calculates location data using location algorithms such as proximity, ranging, triangulation, or signal strength maps. RTLS platforms, like the one introduced in the paper, cover the positioning and middle-ware layers of an LBS.

Once it is clear that an architecture model for RTLS was necessary, the International Standard Organization (ISO) defined two standards: ISO24730-1:2006 [13] and ISO19762-5:2008 [12]. The first defines an Application Programming Interface (API) describing an RTLS service and its access methods, to enable client applications to interface with the RTLS, whilst the second provides a harmonized vocabulary with terms and definitions unique to locating systems in the area of automatic identification and data capture techniques. This glossary of terms enables communication between non-specialist users and specialists in locating systems through a common understanding of basic and advanced concepts.

We can find several examples of location systems with standalone solutions. As we can see, these systems provide particular solutions for an RTLS. Representative examples of them are:

- Authors in [106, 125, 150] take advantage of data coming from only one technology (one type of sensor) for obtaining a monolithic solution.
- Authors in [78, 108] use several technologies, but they were defined for specific applications, in particular for hospitals and museums.
- Other examples of proprietary RTLS with a business model are shown in [8, 22, 24], where the owner companies provide closed solutions for their customers, relying on their own hardware and presenting large numbers of constraints in the location data provided.

The best example of an RTLS platform that works seamlessly with several technologies can be found in the LocON project [86, 87]. This platform provides an interface for the end user that

interacts with an application layer independent of and transparent to the supported technologies [121, 145]. The defined API in LocON is oriented to application support, but it lacks interfaces for users of the platform at other levels of abstraction, i.e. hardware providers or algorithm developers.

3.2 Proposed Architecture Features

Taking into account the previous state of the art, we have considered the following requirements for the proposed architecture that improves the deficiencies of the previous solutions:

- **Technology Independent:** it must support inputs from any kind of measurement (Received Signal Strength (RSS), Time-of-Arrival (ToA), Time-Difference-of-Arrival (TDoA), Angle-of-Arrival (AoA) and raw distance), generic data from any sensor (i.e. inertial sensor such as accelerometer, gyroscope, digital compass, motion detector, image frames from a digital camera, etc.).

  Table 3.1 shows the standardized type of measurements supported by the proposed architecture, to consider any kind of sensor data. The *custom defined data* of a generic sensor can be modelled as an aggregation of parameters of any basic data type: float, integer, strings, boolean or binary data (encoded as string using a method such as Base64, UUEncoding, etc.). Or it can be modelled as an arbitrary long array-list of such kind of parameters. Section 3.5 shows two example case studies and it deeply explains the capabilities of the platform with generic sensors.

RSS	TimeStamp [ms]	AnchorID	MobileID	RSS [dBm]	Seq. Number
TOA	TimeStamp [ms]	AnchorID	MobileID	TOA [s]	Seq. Number
TDOA	TimeStamp [ms]	AnchorID$_1$ AnchorID$_2$	MobileID	TDOA$_{12}$ [s]	Seq. Number
AOA	TimeStamp [ms]	AnchorID	MobileID	AOA [α,β rad]	Seq. Number
DISTANCE	TimeStamp [ms]	AnchorID	MobileID	Distance [m]	Seq. Number
SENSOR	TimeStamp [ms]	SensorID	Custom defined data*		

Table 3.1: Standardized types of measurement supported

- **Multi-Technology:** a target to be located could be comprised of several technologies at the same time. In this way, the requirement of supporting several technologies can be used to merge heterogeneous raw data to provide reliable and precise positions, thanks to the diversity obtained.

  Figure 3.1 shows the UML class diagram of the possible components that are defined in the proposed architecture, needed for considering the previous requirement. As we can see, we have *Nodes* of two types: *Anchor* (with fixed and known positions) or *Mobile* (to be located). The *Networks* are groups of several *Anchor* nodes (at least

one) to form a common WSN. Finally, the *Target* devices are virtual devices with an identifier that aggregate several *Mobile* nodes (i.e. different technologies) and/or generic *Sensors* which can be jointly located with only one system query. As we can see, target nodes can consider multiple technologies, providing diversity that can be exploited by the algorithms.

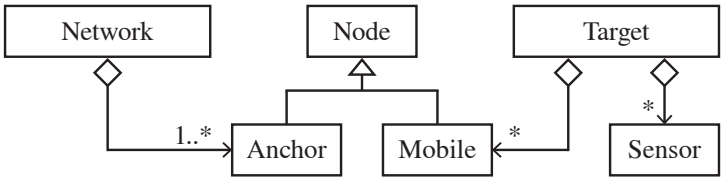


Figure 3.1: UML class diagram of the system nodes, sensors, networks, etc.

- **Multiple coordinate system and map-aware application support:** to give support to the map-aware applications in indoor scenarios, the architecture must be able to work with elements and positions with coordinates relative to different maps, which possibly have different local coordinate systems. The local coordinate systems can be defined by an origin point, a scale, and optionally an associated blueprint bitmap. Moreover, the architecture should support a global coordinate system such as the one used by satellite location systems, and should be able to combine both kinds of coordinate system transparently for users.

- **Data Fusion:** the system must provide mechanisms for data fusion coming from different raw measurements (technologies) and/or positions obtained from different algorithms. On one hand, low-level raw measurement fusion could be considered inside the location algorithms (i.e. Kalman or Particle Filters), since an algorithm can obtain data coming from several technologies (sensors) associated to the same target. And, on the other hand, optional high-level fusion of final estimated positions must be provided directly by the platform. In this way, depending on the LBS considered, client applications could switch between, or combine, different algorithm position estimations.

- **Protection and Security:** the system should support at least one mechanism for secure authentication of external users who want to interact with the system, and optional protection of the data interchanged with the system, for instance by some means of data encryption.

- **API:** the platform must define an API describing the access methods, to enable client applications to interface with the RTLS. This API will define a boundary with syntax and semantics that specifies the seamless interface between external modules and the platform. Since the cited standards ISO24730-1:2006 [13] and ISO19762-5:2008 [12] do

not cover all the requirements of the proposed platform, it is not possible to follow them. However, the API platform is based on them using standardized names and abbreviations for the different actors, commands, modules, parameters, etc.

- **Off-line data:** it must be possible to get measurements and position estimations on-line in real time and also off-line. Therefore, it should be possible to get the latest measurements or positions available or those from specific instants of time (off-line or historical data). Off-line data makes it possible to test and compare algorithms and client applications with a common set of data, and obtain repeatable results which are critical for research purposes.

- **Easy-to-Use:** users without special skills should be able to access the system to perform any of the following tasks:

  - Registration of WSNs, building blueprints, anchor networks, mobile nodes, generic sensors, and insertion of raw measurements into the system. This information will provide an abstraction of the hardware, offering users without hardware skills the possibility to obtain real measurements and sensor data without effort.
  - Obtaining measurements and inserting position estimation information by users who design location and tracking algorithms. They provide positioning data to end-user client applications which do not need special skills in how the location algorithms are implemented, optimized or obtain measurements.
  - Obtaining position information by end-user client applications in real time, for several mobile nodes, sensors, or groups of them, which can be implemented by using different WSN hardware technologies. These users do not need to know how and from which WSNs measurements were taken, nor from which location or tracking algorithms these positions were generated.

3.3 Logic View

Figure 3.2 shows the components that make up the proposed client-server architecture as well as the connections among the components in terms of a logic view:

- **Location Server:** the main component of the hybrid location system, which is composed of several modules for storing measurements and positioning information obtained from external WSNs, sensors, and location algorithms. It allows external actors to insert and retrieve in a flexible way all manner of location information. As a server, it offers an input/output interface to allow remote connections.

- **WSNs and Sensors:** remote client applications that can be deployed in distributed hosts, and which provide all kinds of measurements (as shown in Figure 3.1, RSS, ToA, TDoA, AoA or distance) and generic sensing data (i.e. motion detection, acceleration, turning, inclination, etc.) to other actors such as *Location Algorithms*.

- **Location Algorithms:** client applications responsible for obtaining available measurements and sensing information from the *Location Server* to estimate new positions. Then, they can store the new positioning information inside the system. *Low-Level Fusion Algorithms* (i.e. Kalman and Particle filters) are supported as specialized *Location Algorithms* which would retrieve several kinds of measurements from different WSNs (instead of from only one), accessing the *Location server*.

- **Localization Clients:** client applications that consume positioning information previously generated and filtered for one or more location algorithms, with the option of high-level position data fusion. They can also query for measurements if they need them for a specific purpose.

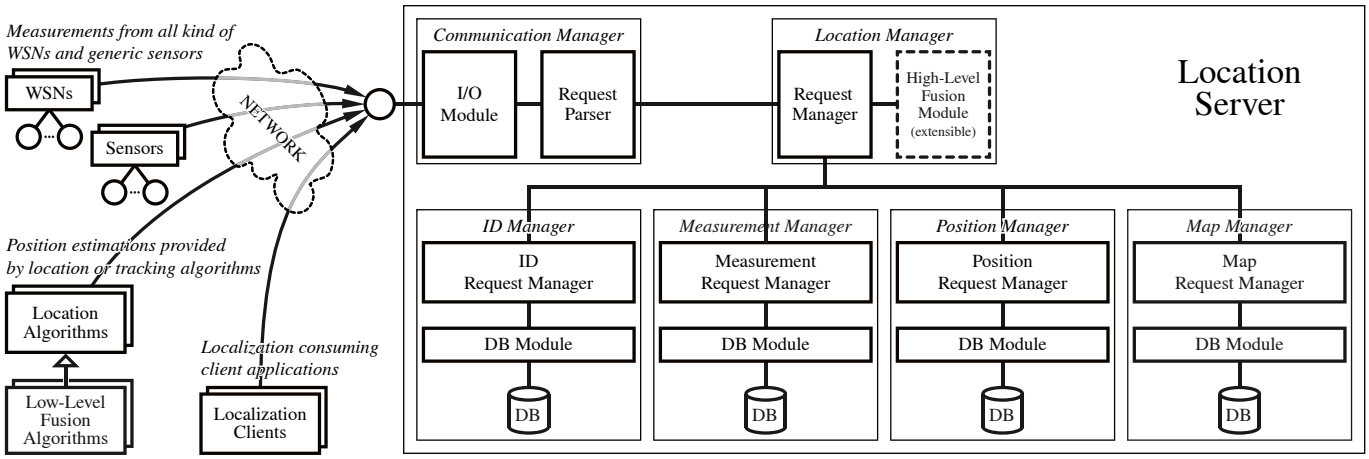


Figure 3.2: Proposed architecture for Hybrid Real-Time Location Systems

3.3.1 Location Server

Figure 3.2 shows a component diagram of the *Location Server* architecture. It is composed of the following subsystems:

- **Communication Manager**: responsible for communications with the server to attend external clients, parsing the received requests and generating appropriate responses. Internally, it is made up of an *Input-Output Module* in charge of the data transmission from or to an external network, and a *Request Parser* module in charge of processing query messages and wrapping output messages. As we will show in the following

sections, these messages are well defined, based on the JavaScript Object Notation (JSON) standard. JSON is a lightweight text-based open standard designed for human-readable data interchange, much more compact and readable than other data formats such as eXtensible Markup Language (XML) [39] and YAML Ain't Another Markup Language (Yaml) [40]. The greater speed and reduced computational complexity of JSON versus other formats has led the main API developments to increase their use of JSON from 6% in 2009 to 20% in 2011, as the only supported formatting system [26]. Seeing the growing inclination of developers and API makers alike for JSON, many well-known companies and Internet services such as Tumbler, Twitter, Facebook, opensocial, Google, digg, Urban Airship, twilio, etc. have decided to eliminate the XML support (when possible) and devote their energies to migrating to a good implementation of JSON [9, 10, 25–27, 56, 143]. Currently there are several implementations of JSON available (i.e. Jackson [48], Jsonlib [49], Google Gson [47], etc.), which even offer cache mechanisms to guarantee maximum performance. Some benchmarks are available in [16–19]. Therefore, we decided to combine JSON with standard TCP sockets to guarantee very low in-system overheads due to message data processing, and to allow a very low latency. In this way we allow any client (even lightweight clients using micro-controllers), using any kind of programming language, to be able to send and receive data with the *Location Server*.

- **Location Manager**: its mission is to analyze all the queries that come into the *Location Server* to find a suitable answer in the server by delegating in the *Position Manager*, *Sensor Manager* or *ID Manager* subsystems. The next element is the *High-Level Fusion Module*. It is responsible for creating fusion data based on the available position estimations from any technology that have been generated by any *Location Algorithm*. This module is easily extensible by a programmer by registering a new fusion module implementation in the system. Adding a fusion method key to an appropriate configuration file and the implementation class should be enough to add a new high-level fusion method to the architecture.

- **ID Manager**: its mission is to store the existing relationships among the *target* devices to be located and the *mobile* nodes and *sensors* assigned to it (see Figure 3.1 for more details about the relationships allowed in the system). This subsystem manages the associations and disassociations among the target devices, mobile nodes and generic sensors, and a log of these associations for future use.

- **Measurements Manager**: receives and stores measurements from WSNs and sensors, and attends queries from external *Location Algorithms* and *Localization Clients*.

- **Position Manager**: its mission is to receive and provide positioning data. Several

Location Algorithms can store estimated positions in the system using this module, while external *Localization Clients* can query these data at any time. The stored positioning data is associated with an identifier of the *target* device or single *mobile* node or *sensor* from where it was measured, and an estimation of its accuracy, which is calculated by means of different aspects such as the kind of technology, the number of sensors available, the number of measurements per time, etc. The associated identifier can be a single mobile device, although more sophisticated *target* devices are managed by the *ID Manager*.

- **Map Manager**: receives and stores coordinate system information associated to several maps. Map-aware elements such as the anchor nodes or the position estimations generated by the *Location Algorithms* (or a specific *Low-Level Fusion Algorithm*) are relative to a particular map registered in the system. The different clients can register new maps, given an origin coordinate, a map scale and optionally a map bitmap (usually the blueprints of a floor of a building for an indoor scenario).

As shown in Figure 3.2, some subsystems are connected to different databases, which can be implemented in the same or in several physical databases. Also note that the *ID Manager*, *Measurement Manager*, *Position Manager* and *Map Manager* use the following two modules: a *Request Manager* to parse the queries and encapsulate the result data, and a *DB Module* to obtain and store information from or into the appropriate database, abstracting and isolating access to it.

When isolating each part within the *Location Server* as we have done, using different modules and layers, we are able to correctly maintain and extend each part in the future. Therefore, we can give a proper support to future application requirements, or adapt the system to specific project constraints with little effort.

3.3.2 High-Level Fusion Module

As we have explained in section 3.2, multiple sensor technologies can be associated to a virtual target device that can be used for its location. With the proposed location system, we can not only query the position estimation of a multi-technology target device with one system call, moreover we can combine all or a subset of technologies grouped by a target device with one system call.

The proposed high-level fusion module of the architecture can combine or select different position estimations available in the system, which were inserted by several location algorithms and which are associated to several sensor networks. This module can automatically switch or combine the available position estimations, and provide an end-user client application with an improved position estimation, also automatically.

The main advantage of the high-level fusion module of the proposed architecture is its extensibility. A software programmer can add a new implementation for this fusion module to the existing ones, adding new capabilities to the system in terms of data fusion. The new data fusion method should be uniquely identified by a string key, which should be added to a configuration file, in addition to other parameters about the path of the new implementation class files, etc.

The proposed architecture message system that will be discussed in Section 3.4 will show how to ask the system for a position estimation, using or not the data fusion capability.

To facilitate the workload of the data-fusion method programmer, the new position estimation message defined in the system (*new_position* message) requires an input position accuracy (*position.acc*) parameter. It is thus mandatory for all the location algorithms to insert an accuracy estimation (in meters) for every position estimation inserted into the system. It is a task of the location system designer to provide this parameter as accurately as possible, through the *new_position* message. Typically based on the specific characteristics of the indoor scenario, the number of anchor nodes available, number of anchors detecting a mobile node, the kind of sensor technology (based on RSS, TOA, etc.), and maybe on an estimation of the variance of the signal level fluctuations, the location algorithm designer should provide the system a dynamic changing *position.acc* value.

As an example, we have considered three different data fusion techniques to illustrate how the proposed *high-level fusion module* works, and how it can be extended. Some of these methods take advantage of the *position.acc* information obtained from the location algorithm estimations, and by means of different techniques for combining or selecting the available position estimations provide filtered position estimations. The three methods implemented are based on classic combination and selection algorithms, typically used to increase the capacity or to stabilize the received signals when using multiple antennas. Note that in this paper we use the same names as those used in the traditional algorithms, but we have adapted them to our location architecture, based on the *position.acc* parameter and the position estimations. In the results section 3.6 we will show the conditions where we gain from using high-level data fusion, compared to the case where we use separate individual WSN technologies.

3.3.2.1 Combination Methods

- **EGC (Equal Gain Combiner)**: identified as `EGC` in the system, this combination method is very simple as it gives equal weight to all the position estimations from the available technologies associated with a single device. It produces the average of all the position estimations, following the equation:

$$pos_{EGC} = \frac{1}{N} \sum_{i=1}^{N} pos_i \tag{3.1}$$

where N is the number of available location algorithms which are associated to a device, and pos_i is the position given by the *i*-th location algorithm. This method does not take into account the *position.acc* parameter.

- **MRC (Maximum Ratio Combiner)**: identified as `MRC`. Unlike the EGC, this method weighs each position estimation depending on its quality, given by the *position.acc* parameter. That is, it gives more weight to the algorithms with the smaller *position.acc*, and is calculated by means of the following expression:

$$\begin{aligned} pos_{MRC} &= \sum_{i=1}^{N} \omega_i pos_i, \\ \omega_i &= 1 - \frac{position.acc_i}{\sum_{j=1}^{N} position.acc_j} \end{aligned} \tag{3.2}$$

Note that in the event of there being only one position available ($N = 1$), the result is directly this position.

3.3.2.2 Selection Methods

- **SC (Selection Combiner)**: identified as `SC`, this method consists in sorting the available position estimations by means of their associated *position.acc* from lowest to highest value, choosing only one value. The goal of this method is to achieve the best position estimation at each instant, without considering the location accuracy fluctuations. In our system we have considered the position estimation associated to the lowest valued *position.acc*.

3.3.3 Location Algorithms and Low-Level Fusion Algorithms Support

As shown in Figure 3.2 any arbitrary location or more sophisticated tracking algorithm is supported. The typical work-flow of the location algorithms is explained in detail in subsection 3.4.3 but basically, they are periodically fed with measurements from a particular WSN, which are stored in the *Location Server*, and they then generate new position estimations in real-time (which are sent back to the Location Server for future use by any *Locatization Client* of other *Location Algorithms*, etc.).

The special tracking algorithms which involve a low-level fusion of raw data from multiple WSN are specialized versions of a *Location Algorithm*, supported in this architecture by the gathering of measurements from several WSNs and sensors through the *Location Server*. The mechanism for obtaining measurements is the same. The multi-threaded implementation of the *Communication Manager* allows for concurrent connections to the *Location Server* and permits

the low-level fusion algorithms to gather the sensor data asynchronously and concurrently at any time, so that they have time to process and merge the data as needed.

Therefore, all kinds of Bayesian filtering algorithm with different observation models are supported within this architecture.

3.3.4 Protection and Security

We considered the OpenID [23] open standard as the most convenient manner of authenticating users with the *Location Server*. It is a highly mature and safe solution, which can be extended in the future. OpenID authentication is now used and provided by several large websites, such as AOL, BBC, Google, Yahoo!, IBM, MySpace, LiveJournal, Facebook, Vkontacte, Steam, Orange, PayPal and VeriSign.

Regarding the confidentiality of data exchanged between the different clients and the *Location Server*, we can optionally add cryptographic support by using security protocols such as Secure Sockets Layer (SSL) [29] or Transport Layer Security (TLS) [31]. These security mechanisms are optional and independent of the rest of the proposed architecture, and they are only implemented for high-level clients with high security requirements. Thin clients such as the ones implemented over a WSN hardware do not usually require such high security which, on the other hand, would increase the network overheads.

3.4 Runtime View

After describing the elements (and interconnections) of the proposed architecture in Section 3.3, in this section we will explain a runtime view which represents the work-flow of the different processes needed by the proposed architecture supporting the capabilities shown in Section 3.2.

These processes are divided into the following four main groups, according to the person responsible for carrying each process. All the processes are considered independent:

- **Network Administration and Measuring Processes**: a network installer has to physically deploy a WSN (of anchor nodes) at an indoor or outdoor scenario, from which to take physical measurements. Therefore, a user registers the *network* elements (*anchors*), *mobiles* and *generic sensors*, all of them with an associated identifier, and the anchors with their real and fixed positions relative to a *map* coordinate system. After that, several external software applications should continuously take measurements from the deployed networks by directly accessing the nodes hardware. They then insert these measurements into the system by creating standardized messages and sending them through a network to the *Location Server*. In this section we will explain in detail the set of standardized commands supported by the system.

- **Target Device Administration**: the same network installer or another user can register virtual *target* devices, which are associated to several *mobile* nodes and/or generic *sensors* used for localization. These targets, and consequently the several elements associated to them, can be queried by other users (in the following steps) with a single system call.

- **Position Estimation using Location Algorithms**: a researcher or location algorithm developer can continuously take measurements from the system, without taking care about how the WSN networks were deployed, or how the measurements were "extracted" from the hardware. These users can implement several variants of a location algorithm which are fed with real measurements, from the same or from different WSNs. Specialized versions of *Location Algorithms* can perform low-level data fusion by retrieving measurements from several *networks* and several *sensors* at a time. After that, they can perform a tracking step, and finally return a position estimation to the *Location Server*. As always, the measurements and positions can be easily obtained from the system or inserted into the system by using standardized commands.

- **Localization Clients**: finally, several end-user client applications can access the system in real time to obtain position estimations from several technologies, optionally using a high-level fusion method registered in the system.

In the rest of this section we will decompose the previous main groups of processes, and explain them in depth. Figure 3.3 shows six groups of processes. As we will explain later, the reason for dividing them into these subgroups is because some of these processes need only be performed once, for instance for initialization purposes.

3.4.1 Network Administration and Measuring Processes

These processes correspond to the work-flow of the group of commands A and B shown in Figure 3.3, and they are explained below:

1. Network Infrastructure Registration

   First, a network installer would physically deploy the anchor nodes of a network (group of nodes with fixed positions) at an indoor or outdoor scenario. Then, he has the option to make the local coordinate system of several floors of a building with their associated blueprints available for future use. He can use the `new_map` command for this purpose, or obtain the maps already uploaded into the system by using the `get_available_maps` command. After that, if this has not been done before, he registers the kind of node technology (i.e. ZigBee, based on RSS measurements or UWB, based on ToA measurements), using the command `new_technology`. This step is performed only once for a specific kind of hardware technology of a network (i.e. ZigBee infrastructure

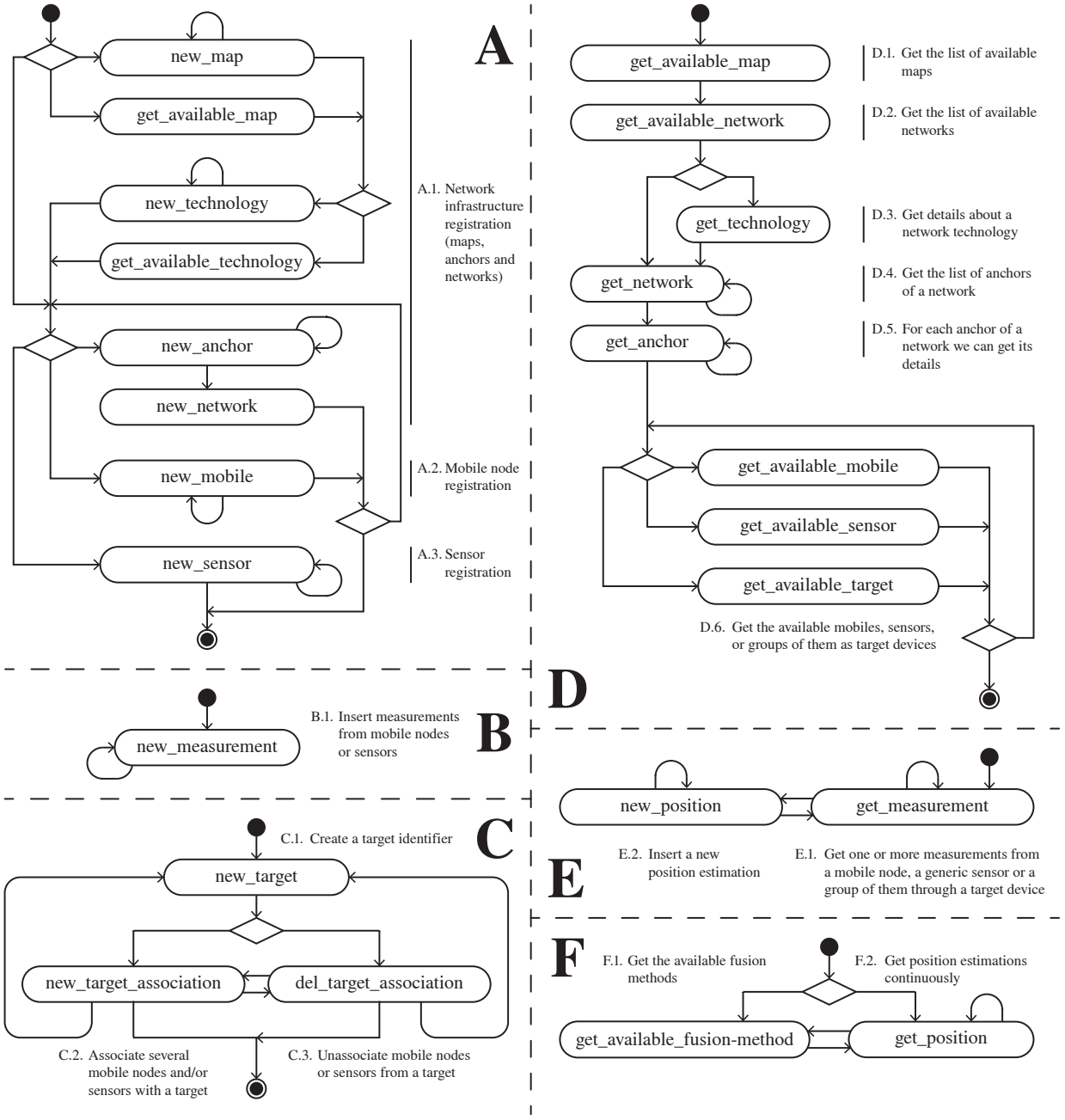


Figure 3.3: Work-flow of the proposed system divided into six groups of messages: A) Infrastructure (maps, anchor and network), mobile node and sensor registration process. B) New measurements from mobiles or sensors. C) Target administration with association to mobiles and/or sensors. D) Location algorithm initialization steps to know the elements available. E) Position estimation process where, a location algorithm estimates new positions after reading measurements. F) The localization clients continuously get position estimations, optionally filtering and merging positioning data.

based on MicaZ [21] nodes working at 2.4 GHz in channel 15, with a specific PAN identifier, etc.). This means that when using another ZigBee platform, for instance based

on Arduino [43] plus XBee [36] nodes, or with the same hardware but configured in another radio channel with a different PAN ID, etc., which is not interoperable with the previous one, the new technology should be registered as a new kind of technology in the system, with a different identifier. He also has the option to retrieve the previously registered WSN or sensor technologies by using `get_available_technology`.

Once the technologies have been registered, the network installer has several alternatives. He usually continues inserting the WSN anchor nodes with their fixed positions into the system (i.e. ten anchors for a desired room, calling the `new_anchor` command ten times). The anchors are associated to a specific technology, which guarantees a type and format of measurements for the system in the future.

Then, it aggregates all these anchors to a same network identifier (which represents a unique WSN) using the `new_network` command. Thus, in the future all of these grouped anchors can be accessed at the same time by a location algorithm. Note that an anchor can only be associated to a single network and that its coordinates are relative to the local coordinate system defined for a specific map (floor or part of a building or other scenario with an origin coordinate, a scale (in pixels/meter) and an optional associated blueprint bitmap image). An example of a `new_map` JSON message is shown in Figure 3.6 in the next section.

As an alternative at the third decision point, the network installer could skip the anchor and network registration, and pass directly to the following mobile node or sensor registration. Note that some previously registered *technologies* only involve generic *sensors* which are not associated to any registered *network* (i.e. inertial sensor, fingerprinting scanpoint, etc.). See group A of messages in Figure 3.3 for more details.

This registration process can be repeated for a different kind of node platform and networks as many times as needed. Other users who have already implemented some location algorithms and tested them with simulated measurements can now test their algorithms with real measurements, without having to be skilled in either hardware deployment or low-level programming of access to the WSN hardware in order to extract physical measurements.

2. Mobile Node Registration

   The network installer also registers new mobile nodes in the system for use by other users with no hardware skills. He uses the `new_mobile` command.

3. Generic Sensor Registration

   The final registration step is for generic sensors such as accelerometers, gyroscopes, digital compass, etc., which can give localization support to a tracking algorithm, for

instance. They must be registered at this step using the `new_sensor` command. Note that a customized kind of technology must have previously been registered in step A.1 for each generic sensor. For instance, for a 3D accelerometer we would require three float values associated to each measurement, representing the acceleration on the three axes. Each sensor measurement format is unique and can be custom defined as shown in Figure 3.1, as any combination of data types supported by the system. The system will use this kind of custom technologies so defined to parse and verify the format of new measurements to be inserted into the system in the future.

1. Measuring Process

   The next two measuring process steps can be performed by the same network installer as in the previous registration steps or by another user, who is familiar with the deployed hardware platform, and who knows how to extract physical measurements from the hardware. This user should implement a piece of software that continuously accesses one or several anchor nodes at low-level (which can be quite sophisticated depending on the kind of technology) to be able to communicate or somehow detect several mobile nodes. This software finally extracts physical parameter information like RSS, TOA, AOA, etc., or generic sensor data (for generic sensors such as accelerometers, gyroscopes, etc.). It should use the `new_measurement` command to insert new measurements into the system, and make them available for the other users (i.e. the location algorithm designer or an end-user location application).

3.4.2 Target Device Administration

As explained in Section 3.2 we can group and identify several *mobile* nodes and generic *sensors* by a single and unique *target* device identifier. In this step, somebody in charge of the device administration can create new virtual target devices and associate or disassociate mobile nodes or generic sensors with or from it. The `new_target` command can be used to register a new target device identifier in the system, and then the `new_target_association` and `del_target_association` commands can be used to add or to delete mobile nodes and sensors to the target device. This process corresponds to the work-flow and the group of commands C shown in Figure 3.3.

3.4.3 Position Estimation using Location Algorithms

The researcher or user who implements a location algorithm is the next actor in the system. He would typically be a user who has already implemented a more or less complex location or tracking algorithm, and who probably uses a simulation environment such as Matlab and makes use of simulation measurements to test his algorithms. This user can now feed his algorithm

using real measurements from one or several WSNs, with the advantage that he does not have to spend any time deploying a WSN or accessing the hardware. He can directly query which *technologies* are available (registered) in the system, where the *networks* (groups of *anchors*) are physically deployed (relative to a *map*), with which *technology* a particular *network* (or kind of *sensor*) is implemented, which *mobile* nodes and *sensors* are available for experiments, etc. Finally, he can query for several on-line or off-line measurements from several WSNs. All these tasks can be easily performed using standard commands as always, with no need for advanced skills in hardware or software implementation. In section 3.5 we will explain how a researcher could connect the *Location Server* to take real measurements.

This process corresponds to the work-flow and the groups of commands D and E shown in figure 3.3. The first part (group D of commands), as an initialization step, is only performed once to know which *maps*, *technologies* and *networks* are available, where the *anchor* nodes are deployed, and which *target* devices, *mobile* nodes or *sensors* are available for testing. Remember that the networks and other nodes were probably deployed and registered by another user. Taking this information into account, the researcher can now configure his location algorithm parameters, and enter the place where the network is deployed (i.e. in a building) to make measurement studies and test his algorithms with real measurements.

Once his algorithms has been initialized with the information for a specific network, the researcher should start getting measurements from the system in a continuous fashion, and then generate position estimations based on them. This second group of commands is depicted as E in figure 3.3. Finally, he will register these positions in the system, making them available to other *Localization Clients* who do not have skills in location algorithms and only want to obtain final position estimations.

3.4.4 Localization Clients

Finally, a localization-consuming client application wants to remotely query the position estimation of a target device (that can be multi-technology) or of a single mobile node or sensor. This user does not need to have any skills in how the WSN was deployed, how the measurements were extracted from the hardware or how the positions were estimated by one or several location algorithms.

This process corresponds to the work-flow and group of commands F shown in figure 3.3. Firstly, the user can optionally ask for the available high-level fusion methods registered in the system, to later obtain position estimations using data fusion. Or he can directly and continuously query the system for position estimations for a specific target device, mobile node or generic sensor. Note that he could also perform some initialization steps as in the group of commands D, to know the maps, networks, targets, mobile nodes and sensors available.

To summarize this section, figure 3.4 shows the set of the messages that all the actors would

continuously use after their initialization steps.

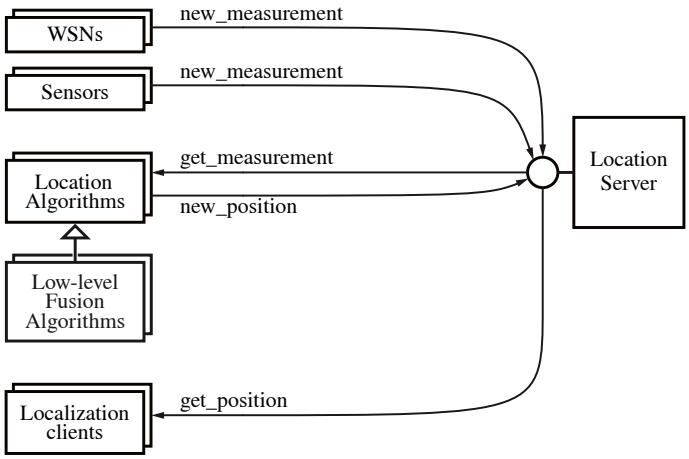


Figure 3.4: Once the network infrastructure is registered, and the location algorithms and clients configured, these actors continuously use the same type of messages to save and retrieve data with the system.

Note that we do not show other messages to modify and delete the information available in the *Location Server*. Commands such as `set_map`, `set_technology`, `set_anchor`, `set_network`, `set_mobile`, `set_sensor`, `del_map`, `del_technology`, `del_anchor`, `del_network`, `del_mobile`, `del_sensor`, `del_target`, `del_measurement` and `del_position` are also available to privileged users. And we do not merely support a global coordinate system for anchors and position estimations, but also relative position coordinates inside a building, covering all the common scenarios.

3.5 Implementation Example Case Studies

In this section we will explain two implementation example case studies to better understand the capabilities of the system regarding the fusion of multiple technologies, the mix of mobile nodes and generic sensors, how to define new kind of sensor technologies, etc.

3.5.1 ZigBee and UWB Networks, plus an Inertial Sensor (Accelerometer)

In the first case, we have chosen the ZigBee and UWB technologies plus some inertial sensors (acting as generic sensors) to better explain how the commands described in the previous section 3.4 are formatted, in the case of having a multi-technology target device to be located. Moreover, we will make it clear how to call the system to obtain high-level fusion position estimations in this case.

The ZigBee network could be implemented with MicaZ [21] nodes working at 2.4 GHz. In this example we will have four anchor nodes `zb_A1` to `zb_A4` and one ZigBee mobile node `zb_M1`.

The UWB network could be implemented with the PLUS platform of TimeDomain [24]. In this example we will have four anchor nodes (UWB readers `uwb_A1` to `uwb_A4`) and one UWB mobile node (UWB tag `uwb_M1`).

Additionally, we have decided to take the measurements from the accelerometer (`accel_1`) integrated in a smartphone to illustrate how the measurements of any generic *sensor* could be formatted and processed.

The ZigBee and UWB mobile nodes, plus the accelerometer, will be associated together by a unique *target* device identifier, in order to be conveniently located with a single system call. Figure 3.5 shows how we associate a unique identifier to the virtual target device associated to both mobile nodes and the sensor. Note that all these pieces of hardware should be attached and moved together in order to be tracked correctly. In a real case, they would all be integrated in a multi-technology portable device.

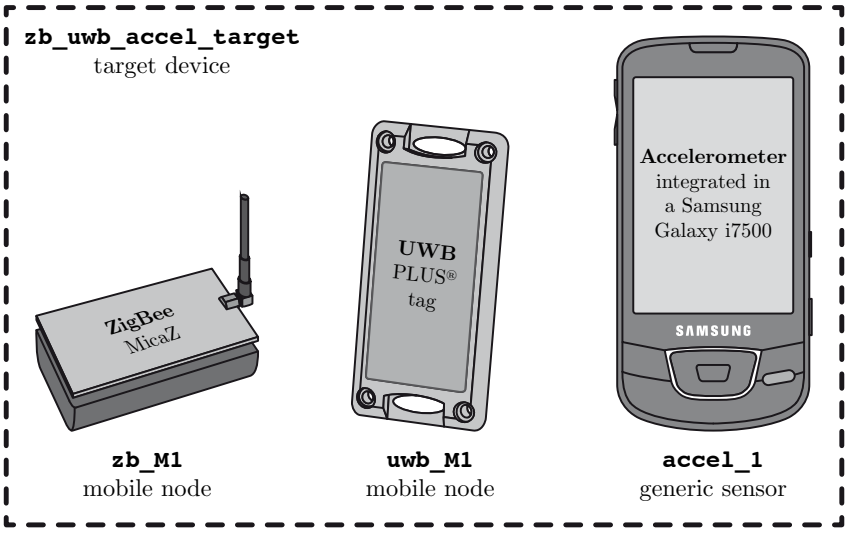


Figure 3.5: The `zb_uwb_accel_target` identifier represents a virtual target device, and it is associated to the `zb_M1` and `uwb_M1` mobile devices and to the `accel_1` generic sensor.

In order to save space in this section, we will usually refrain from showing some kinds of messages twice, for the cases where the ZigBee and UWB messages are somehow identically formatted. We will also usually refrain from showing the reply commands obtained from the system, when they are simply OK acknowledgement messages, as they actually are in most of the cases. We always assume that there will be no communication problems with the location server.

Next, in this section we will explain the four kind of processes introduced in section 3.4, applied to the example case study of using ZigBee, UWB and an accelerometer, by using similar

flowchart diagrams but with specific JSON messages for each entity.

As in Figure 3.3 in Section 3.4, groups A and B of messages in Figure 3.6 show the network administration and measuring processes in detail. For the measuring process we assume that we have already implemented three pieces of software for obtaining measurements from the ZigBee, UWB and accelerometer hardware. On the one hand, a client collects RSS data coming to a PC by USB port, from a ZigBee gateway (or a sniffer). On the other, another client collects TOA measurements from the TimeDomain UWB platform. With the software API provided by TimeDomain it is quite straightforward to implement a client that takes raw TOA measurements from the hardware and optionally transform them into TDOA measurements (when needed, grouping the anchor TOA messages in pairs). These two pieces of software that abstract the communications with the WSN hardware have to insert the new RSS and TOA measurements (when they are available) into the *Location Server*, by using the `new_measurement` command (as shown in the first message of group B in Figure 3.6). Finally, a small piece of software, implemented in Android for example, could collect the inertial information from its accelerometer by a predefined sampling period of time, and periodically send it to the *Location Server* by also using the `new_measurement` command (with different parameters).

The target device administration is explained by group C of messages in Figure 3.6.

The position estimation using location algorithms is shown in group D of messages in Figure 3.7. We show several commands for the initialization of the location algorithms, while in group E of messages we illustrate the loop process of obtaining measurements from the system (for a desired target, mobile node or sensor) and inserting new position estimations. In this example, two location algorithms should be implemented in order to process the RSS and TOA measurements separately. Usually the location algorithms are implemented by a researcher using a high-level simulation environment such as Matlab, and he only has to add some functions to send and receive JSON messages using TCP/IP sockets. This is quite straightforward using a Java client for processing JSON messages, as Matlab is completely integrated with the Java Virtual Machine (JVM).

As explained in Section 3.2 we wanted a highly flexible system for obtaining both measurements and positions. Our implementation allows to obtain both on-line and off-line measurements in real time, asking in different ways:

- We can ask for a list of the latest N available measurements stored in the server DB.

- Asking for the available measurements in the last S seconds, starting the count from the time when the call message is received by the server. Note that we can retrieve a greater or smaller number of measurements, depending on the sampling period of the WSN measuring client.

- Or we can ask for off-line measurements registered in the past. This is a flexible manner to test location algorithms using the same campaign of measurements in each case, to produce repeatable results. The simulation results that are explained in section 3.6 are obtained using this strategy. Note that we can insert both real or simulated measurements

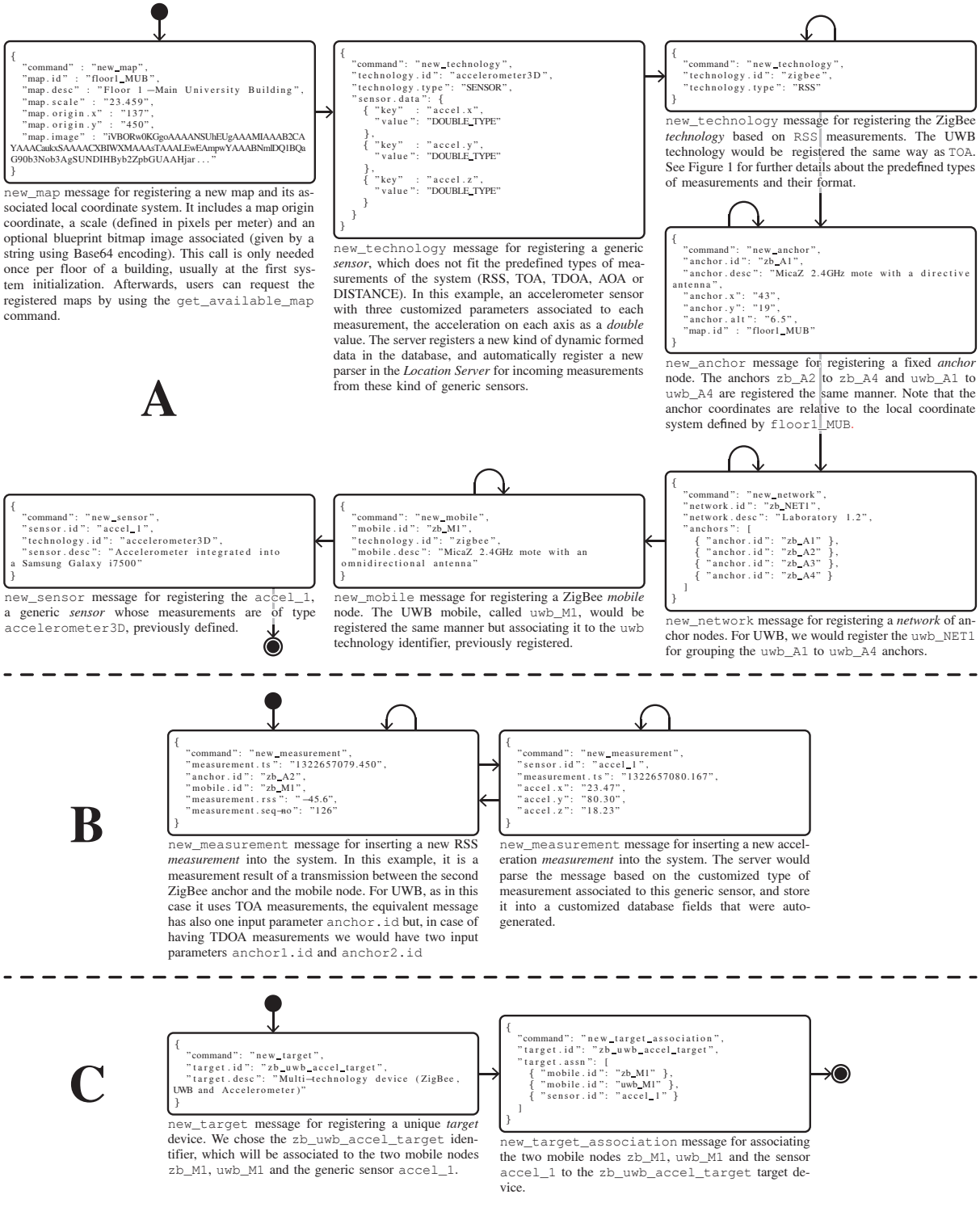
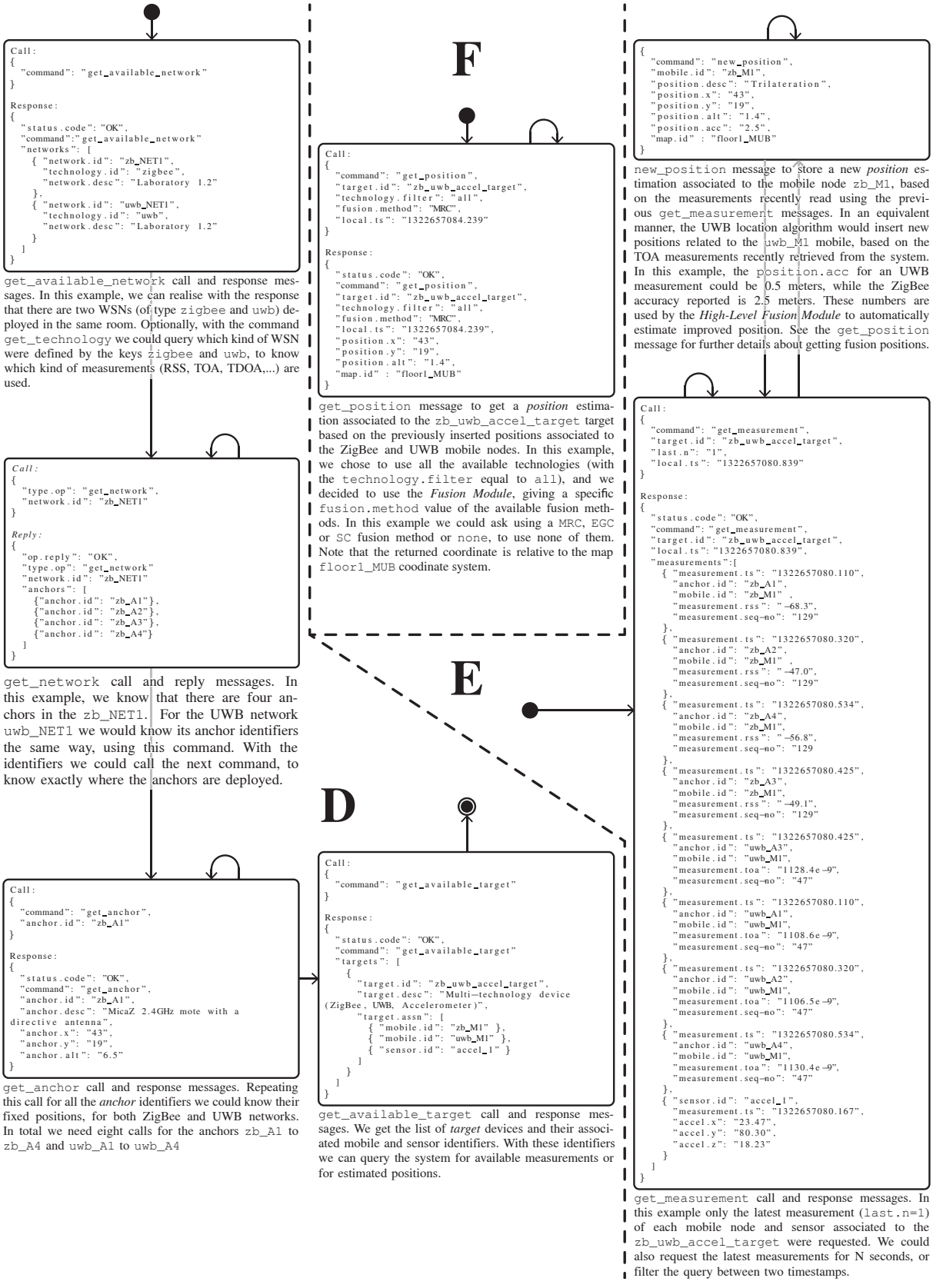


Figure 3.6: Groups of messages A, B and C



get_available_network call and response messages. In this example, we can realise with the response that there are two WSNs (of type zigbee and uwb) deployed in the same room. Optionally, with the command get_technology we could query which kind of WSN were defined by the keys zigbee and uwb, to know which kind of measurements (RSS, TOA, TDOA,...) are used.

get_network call and reply messages. In this example, we know that there are four anchors in the zb_NET1. For the UWB network uwb_NET1 we would know its anchor identifiers the same way, using this command. With the identifiers we could call the next command, to know exactly where the anchors are deployed.

get_anchor call and response messages. Repeating this call for all the *anchor* identifiers we could know their fixed positions, for both ZigBee and UWB networks. In total we need eight calls for the anchors zb_A1 to zb_A4 and uwb_A1 to uwb_A4

get_position message to get a *position* estimation associated to the zb_uwb_accel_target target based on the previously inserted positions associated to the ZigBee and UWB mobile nodes. In this example, we chose to use all the available technologies (with the technology.filter equal to all), and we decided to use the *Fusion Module*, giving a specific fusion.method value of the available fusion methods. In this example we could ask using a MRC, EGC or SC fusion method or none, to use none of them. Note that the returned coordinate is relative to the map floor1_MUB coodinate system.

get_available_target call and response messages. We get the list of *target* devices and their associated mobile and sensor identifiers. With these identifiers we can query the system for available measurements or for estimated positions.

new_position message to store a new *position* estimation associated to the mobile node zb_M1, based on the measurements recently read using the previous get_measurement messages. In an equivalent manner, the UWB location algorithm would insert new positions related to the uwb_M1 mobile, based on the TOA measurements recently retrieved from the system. In this example, the position.acc for an UWB measurement could be 0.5 meters, while the ZigBee accuracy reported is 2.5 meters. These numbers are used by the *High-Level Fusion Module* to automatically estimate improved position. See the get_position message for further details about getting fusion positions.

get_measurement call and response messages. In this example only the latest measurement (last.n=1) of each mobile node and sensor associated to the zb_uwb_accel_target were requested. We could also request the latest measurements for N seconds, or filter the query between two timestamps.

Figure 3.7: Groups of messages D, E and F

to test a location algorithm by sending `new_measurement` commands through the network. The mechanism for inserting the measurements into the system is the same in both cases, using the `new_measurement` command filled in with simulated or real data. Therefore, for testing purposes we can also take advantage of this feature, and simulate thousands of nodes and large numbers of location algorithms accessing the system at the same time. This allow us to perform stress tests, measure latency versus load, check the performance of disk access to the DB under certain controlled conditions, etc.

Finally, the *Localization Clients* can use the `get_position` command to obtain position estimations from a remote computer or device, via LAN or Internet connection. As shown in group F of messages in Figure 3.7, and similarly to the case of `get_measurement`, they can request on-line and off-line position estimations: the latest N known positions, the latest positions stored for S seconds starting from the present, or a list of positions between two epoch timestamps.

Moreover, they can filter by technology when asking for a target device position, to use all or a subset of the mobile nodes and sensors associated to a target device. And finally, they can optionally make use of a high-level position fusion method implemented in the *Location Server*, such as the EGC, MRC or SC proposed in this article for exemplifying purposes. See section 3.3.2 for more details about the fusion methods implemented.

3.5.2 WiFi Fingerprinting

In this second example case study we will show how to apply our architecture in a real indoor scenario when using RSS measurements and a fingerprinting location algorithm [70, 110].

We take advantage of the WiFi access-points already deployed in a building acting as beacon transmitters. We use a WiFi tag that during the *off-line* or *calibration* phase acting as an access point (AP) scanner, measuring at different locations of a building. The Received Signal Strength Indicator (RSSI) from several APs is measured at chosen locations, called *scan points* (SP) or *calibration points*. These measurements are called *fingerprints* of the SPs and they are part of a calibrated *radio map* [70, 109, 116, 124]. Then, during an *on-line* or *location estimation* phase of the algorithm, the WiFi tag acts as a target sensor, detecting the nearby APs while it moves. For simplicity, in this example we assume that the state of the target includes only its location (x and y coordinates), and we associate the SP to a zone identifier of the building for convenience.

We have used an Android smartphone with integrated WiFi connectivity to implement both roles during the off-line and on-line phases, as shown in Figure 3.8.

First we specifically define the data structures for storing the measurements and the fingerprints associated to each SP. Conceptually we have chosen the most complete way to represent this information as raw lists of RSS measurements associated to an arbitrary number of surrounding APs. This way the more or less complex fingerprint algorithm can use not only

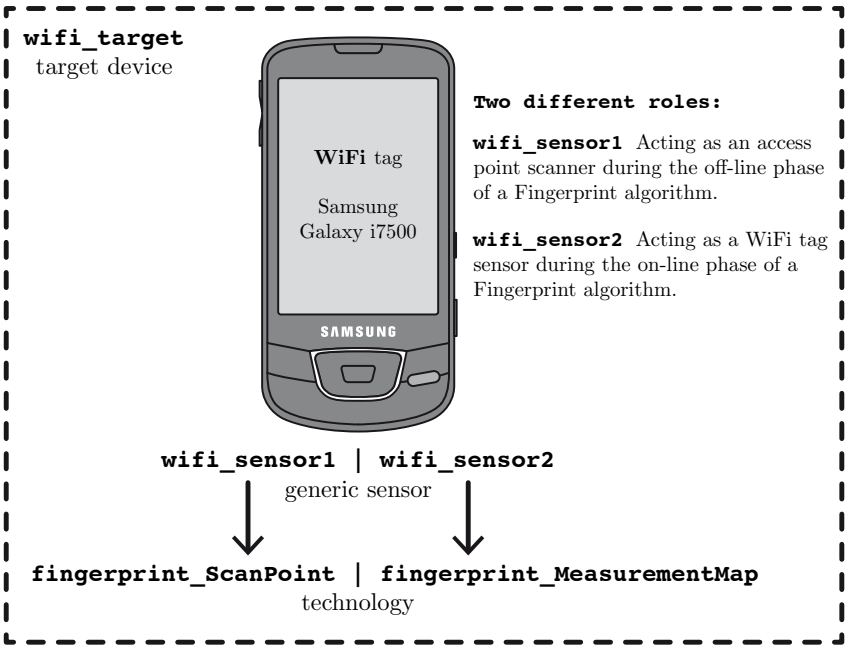


Figure 3.8: The wifi_target represents a target device associated to the wifi_sensor1 or wifi_sensor2 implemented with the same physical smartphone device, in the example.

the mean [70], mean and variance [110], number of available RSS measurements associated to an AP identifier, and also the whole list of measurements to define a histogram [135].

Figure 3.9 shows a UML class diagram representing the data structures needed. On the other hand, we have defined a `RssVector` class which makes it possible to store an arbitrary long RSS list of measurements, which is associated to a specific AP by using a hash-map `MeasurementMap`. We can add new RSS measurements to a `MeasurementMap` for an specific pair (`apId-rss`), by calling its `addMeasurement(apId, rss)` method. It first checks if there are previous measurements associated to the `apId` by searching in the list of available keys, and then it decides whether or not to call the `addMeasurement(rss)` method of its associated `RssVector`. Otherwise, it creates a new key-value entry in the `MeasurementMap` with the key `apId` associated to an empty `RssVector`.

On the other hand, a fingerprint associated to an SP is modelled with the `ScanPoint` class. The coordinates of the SP are stored in the `position` object (of kind `Position`). We also have some attributes such as `ts` (to store the timestamp of the beginning of the fingerprint), a `duration` (in seconds, to store how long we have measured in the said SP), a `zoneId` (zone identifier to associate the fingerprint to a specific zone of a building, for convenience), a `tagId` (tag identifier used for calibration) and, finally, a `measurements` attribute associated to a `MeasurementMap` hash-map object which contains the whole fingerprint measurements.

Finally, the `FingerPrint` class has a static method to directly run the fingerprinting algorithm from any client without instantiating it. In this example we implemented a K-Nearest Neighbour (KNN) fingerprint algorithm [116, 137] as it is a commonly used fingerprint

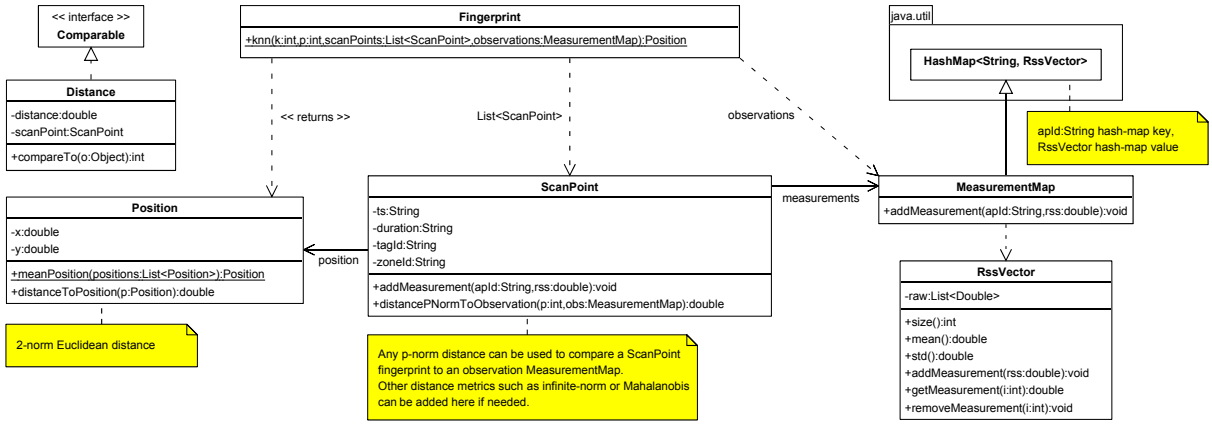


Figure 3.9: UML class diagram representing how to implement any Fingerprinting algorithm in any language (e.g. a KNN programmed in Java).

algorithm in the literature. As we can see, with the `k` parameter we specify how many K "nearest" neighbours coordinates are used to average the position estimations. With a parameter `p` we choose the p-norm distance metric used to compare each `ScanPoint` fingerprint with an observation `MeasurementMap`. And finally we have to give as parameter the list of `ScanPoint` (of a specific radio map) and an observation `MeasurementMap` hash-map with the latest observed RSS measurements.

Following once again the work-flow process explained in Section 3.4, we have to prepare the server for the available target and sensors, and register the new kind of sensor data that the WiFi sensors will send to the *Location Server*. In this case, we have two new kind of sensor measurements to represent a fingerprint associated to an SP (during the off-line phase) and an observation from multiple APs (during the on-line phase).

In Figure 3.10 we show a flowchart diagram with some of the custom defined JSON messages. During the network and target device administration processes we have to define the two kinds of *technologies*, then register the two WiFi sensors associated to each of the technologies, and finally associate both sensors to a unique target device for convenience, to facilitate the work of obtaining positions at a higher level for other clients. A localization client can easily ask for the `wifi_target` positions. During the measuring process we have to give measurements to the system with the `new_measurement` command: fingerprints during the off-line phase, and observations during the on-line phase, as shown in Figure 3.10.

After this point, we have to implement two pieces of software into our WiFi tag device (in our example an Android smartphone):

- To support the first role of the tag as an AP scanner during the off-line phase of the fingerprinting algorithm, we have to implement an application with a more or less complex Graphical User Interface (GUI) where we can fill the *position* coordinates (x,y)

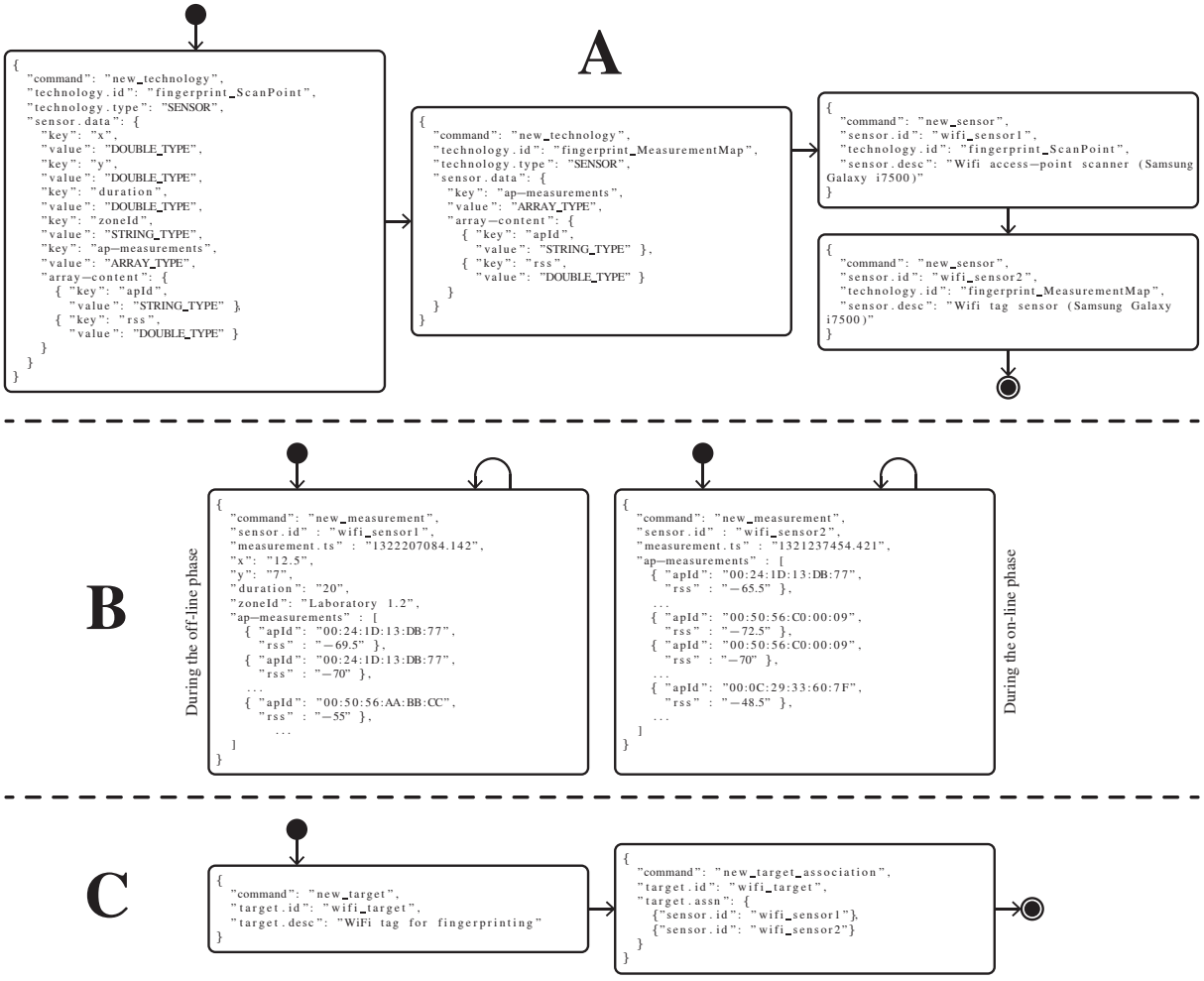


Figure 3.10: Groups of messages A, B and C of the fingerprint case study

of the SP, the *duration* of scan, and a *zone identifier* from where we will scan[1]. After pressing a "Start" kind of button of the GUI the application will call the WiFi API of Android to retrieve the surrounding AP measurements during few seconds, using the WiFi interface in monitor mode by detecting RSSI levels of the surrounding APs during a scan window on all or a subset of the WiFi channels. The application in this example can easily be implemented in Java (as Android uses Java) using the same data structures as defined in Figure 3.9. When the "Start" button is pressed we also start a timer and the application annotates the current system time as a UNIX timestamp. Once a timer exceeds the given *duration* time, the program will stop feeding the *ScanPoint* class with new measurements (pairs apId-rss), and it will send the whole fingerprint associated to the SP to the *Location*

[1]Another possibility would be to take advantage of the maps already registered in the system, retrieving all or some of them through the network using the `get_available_map` command, and performing the same kind of tasks: show the different scanpoints positions directly on each map in the GUI, allow the user to click on a relative coordinate of a map, etc.

Server by using a single JSON message `new_measurement`, putting its own *sensor.id* to "wifi_sensor1", the timestamp to the current time of the system, and the remaining parameters according to the definition of `fingerprint_ScanPoint` technology. See Figure 3.10 for further details about the JSON messages. This process can be repeated as many times as needed, moving the WiFi tag to another position of the building, updating the current scanning coordinate on the GUI, pressing "Start" again, and so on.

- To support the second role of the tag as RSS scanner during the on-line phase of the fingerprinting algorithm, we have to implement a second application that could run in the background or not, which never stops sending measurements to the *Location Server*. In a non-ending loop it calls the WiFi API of Android to scan different channels and after a predefined period of time (i.e. 2 s) it packages all the available observations to the server using the `new_measurement` command, putting its own *sensor.id* to "wifi_sensor2", the timestamp to the current time of the system, and the remaining parameters according to the definition of `fingerprint_MeasurementMap` technology.

On the other hand, a location algorithm researcher can execute a KNN fingerprinting algorithm (for example implemented in Matlab) to estimate positions of the WiFi target. First, he initializes it using several fingerprints associated to the SPs, i.e. *fingerprint_ScanPoint* data, which can be retrieved from the *Location Server* using the `get_measurement` command. Then he gets real observations at a certain sampling speed, i.e. *fingerprint_MeasurementMap* data, again using the `get_measurement` command. After each position estimation is calculated by the fingerprinting algorithm, he can optionally decide to filter the new position estimation based on previous position and a map cartography. And finally, he has to send each position estimation to the *Location Server* using the `new_position` command. This process is repeated continuously.

Finally, one or several *Localization Clients*, as in section 3.5.1, can ask for the position estimations of the `wifi_target` and show the information in a GUI application, a webpage, or wherever they want.

3.6 Validation Results

Up to this point we have shown how different actors involved in a location system would interact with a *Location Server* by sending and receiving JSON messages through a LAN or Internet network, when we have different WSNs and devices to be located, different fusion methods available, etc.

In this section we present some validation results by considering virtual hardware nodes whose measurements will be simulated by software. Only the measurements are simulated

based on propagation models and real scenario noisy conditions. The rest of the code is common for a real implementation, and we send and receive messages with the *Location Server* through a real network. Therefore, using the same software to work with real measurements is straightforward. Also note that with a simulated scenario we can perform stress tests with the server, evaluate the performance of a huge amount of concurrent connections, simulate many kinds of sensors available at the same time, various fusion methods, etc.

The simulated scenario consists of two ZigBee and UWB WSNs deployed in an indoor scenario. We will track the position of two mobile devices (a ZigBee node and a UWB tag) testing the three illustrative high-level fusion methods shown in Section 3.3.2. We make use of the `LocationClient` Java client partially introduced in the previous section, and from Matlab we simulate a moving target device associated with the two mobile devices of the ZigBee and UWB networks, which moves around a large room where both networks were deployed. This Java client sends real messages through the network between a computer with Matlab (acting as a client) and another computer where we deployed the *Location Server* software plus a MySQL database. The server is implemented using Java language and it supports high-performance mechanisms as a pool of threads to attend concurrent TCP socket connections and a pool of connections to the MySQL database in order to support hundreds of concurrent requests from the network. In this way the system is scalable to support even thousands of concurrent requests.

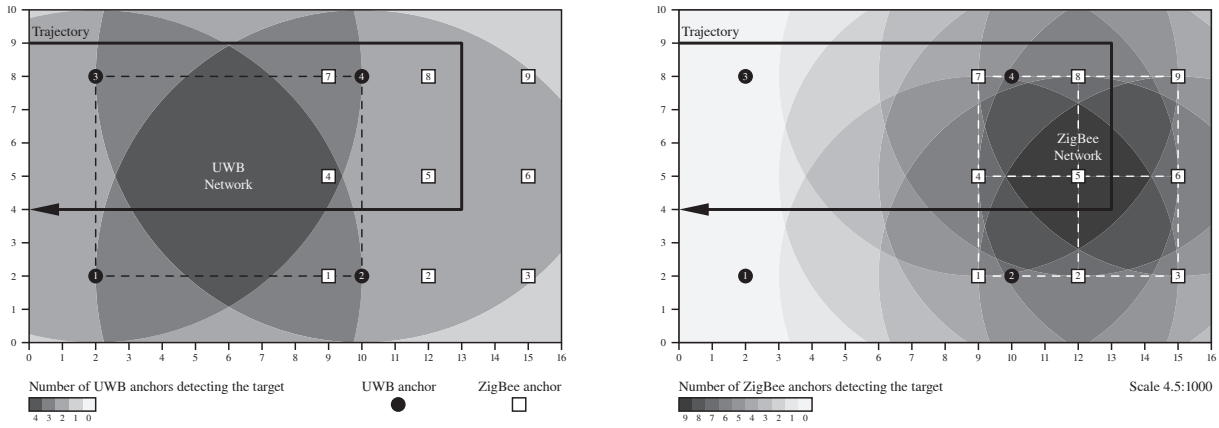
We simulated the different actors of the location system:

- **The WSNs client:** Instead of getting real measurements from a hardware network, we will simulate RSS and ToA noisy measurements, based on noise levels that typically appear in real indoor scenarios [138]. Each $T_s = 0.5\,\text{s}$ we will have a new measurement when a mobile node is inside the coverage area of an anchor node of the same technology. On each iteration we will send each measurement to the server through the network using the `new_measurement` command, as would be done with real measurements obtained from a real WSN (when they are available).

- **Location Algorithm:** The second actor of the system is the location algorithm responsible for estimating new positions based on the measurements obtained from the system, and previously introduced by the WSNs clients. In this case we make use of the `get_measurement` and `new_position` commands to retrieve measurements and store new positions in the server. We will explain the location algorithms used in more detail below.

- **Location client:** The last actor will track the positions generated by the location algorithms by using the `get_position` command, querying the ZigBee and UWB nodes separately and together, using the three high-level fusion methods proposed.

Figure 3.11 shows the indoor scenario where the two WSNs are deployed. There are four UWB anchors deployed on the left-hand side of the 16×10 meter scenario, with a simulated coverage of eight meters, and nine ZigBee anchors deployed on the right-hand side of the room, with a simulated coverage of six meters. With different grey scale colours we represent the coverage areas where we will be able to track the UWB and ZigBee nodes respectively. Note that we will need measurements from at least three anchors at the same time to be able to estimate a new position.



Time [s]			Position [m]		
0	-	26	(0,9)	→	(13,9)
26	-	36	(13,9)	→	(13,4)
36	-	62	(13,4)	→	(0,4)

Figure 3.11: Simulated indoor scenario with two ZigBee and UWB networks deployed

The networks are partially overlap to simulate the effect of having coverage with only one of the networks, then with both, then again only with the second network, and so on. We will analyze the effect of tracking each mobile node separately, versus the use of both technologies together by means of the high-level fusion module available in the *Location Server*.

The trajectory is quite simple, going from one side of the room to the other, turning twice to the right and coming back again, crossing through the area of maximum coverage of both networks. At the bottom of Figure 3.11 we show the time spent in walking one round of the trajectory at a speed of 0.5 m/s. In total 62 s are needed to perform one round.

For the UWB network we assume we have TOA measurements affected by Gaussian noise with zero mean and 1 ns of standard deviation, while in the case of ZigBee we assume an exponential log-normal path-loss propagation model [127] with a Gaussian noise of zero mean and 3 dBm of standard deviation. The path-loss exponent was assumed as known and constant (equal to 1.8) throughout the simulation. Therefore, we assume that the received signal strength decays with the power of 1.8 each meter after a reference distance considered equal to 0.5 m.

Since the aim of this article is not to compare algorithm performance we have considered

the well-known Non-linear Least Squares algorithm [146][2] for both technologies, ZigBee and UWB. With this algorithm we can minimize the sum of the squares of the distance errors to the anchors by minimizing the following objective function:

$$\begin{aligned} F(x,y) &= \sum_{i=1}^{N}(\overline{r}_i - r_i)^2 = \sum_{i=1}^{N} f_i(x,y)^2, \\ f_i(x,y) &= \overline{r}_i - r_i = \sqrt{(x-x_i)^2 + (y-y_i)^2} - r_i \end{aligned} \tag{3.3}$$

where $\overline{r}_i$ is the exact radio of the circumference centred in the i-th anchor, r_i is the noisy radio of the i-th anchor, and (x_i, y_i) is the known position of the i-th anchor.

To be able to properly use the high-level fusion module of the *Location Server*, the location algorithm designer must provide an appropriate position accuracy (*position.acc*) value inside each new position estimation (as a parameter in each `new_position` command). After performing one survey round and observing the instant error for both ZigBee and UWB cases, we have chosen the following *position.acc* formulas for the simulations:

$$\text{position.acc}_{ZB} = 1 + 0.042(\text{detecting\_anchors} - N)^2 \tag{3.4}$$

$$\text{position.acc}_{UWB} = 0.5 \tag{3.5}$$

where N is the total amount of ZigBee anchors, and *detecting_anchors* is the number of ZigBee anchors detecting the mobile node. With the first quadratic formula, we have a maximum accuracy of 1 m when there are nine active ZigBee anchors, and a minimum accuracy of about 2.5 m when there are only three anchors covering the mobile node[3]. For the UWB case, we fix the accuracy to 0.5 m in all cases, regardless of the number of UWB detecting anchors. Thus for instance, the SC fusion method presented in section 3.3.2 will always select the UWB positions when both ZigBee and UWB technologies are covering the target at the same time, as UWB's accuracy is always better than ZigBee's.

Figure 3.12 shows the instant error[4] of the simulated trajectory, after repeating it ten times to average the results (the ZigBee measurements are quite noisy). Before each round starts, we always initialize the position estimation of the location algorithm at the $(0, 9)$ coordinate. In the sub-figure on the left we show the case when the ZigBee node is used alone. In red we show

[2]We used the Nonlinear Least-Squares algorithm available in Matlab using the `lsqnonlin` function of the *Optimization Toolbox*.

[3]The decision to choose an accuracy bound between 1 and 2.5 m is simply a possibility that the location algorithm designer should choose according to the scenario and available number of anchors. In this case a separation of 3 m between ZigBee anchors lets us choose a lower bound of about 2.5 m (when *detecting_anchors* = 3), while in the best case of having nine anchors detecting the target, uncertainty could reach the 1 m level (in this case the right hand side of equation 4 would become zero as *detecting_anchors* = 9 = N). Therefore, the constant 0.042 was chosen to reach the 2.5 m lower bound, and should be adjusted, or the whole position.acc$_{ZB}$ calculation could be replaced by another formula.

[4]The instant error is calculated by the Euclidean distance (2-norm) between the real position of the target and the estimated position (calculated by the LS algorithm) at each time instant.

the instant error, which is the absolute difference between the current estimated position and the real position of the ZigBee node. The first slope from 0 to about 6 meters of error is due to the lack of ZigBee coverage on the left-hand side of the room until the ZigBee anchors 4, 7 and 8 cover the mobile node (until it reaches the $(6, 9)$ coordinate). Then, from this instant ($t \simeq 12\,\text{s}$) until it loses coverage again ($t \simeq 50\,\text{s}$), the ZigBee mobile node is detected from between 3 and 9 anchors at any one time. With a blue line in the figure, we represent the number of activated anchors with respect to time, and with a green line the calculated position accuracy defined in (3.4). Note that the *position.acc* is drawn as zero in the figure when there are no positions available. In fact, there are no positions stored in the server for such time intervals and, therefore, they have no associated *position.acc* value.

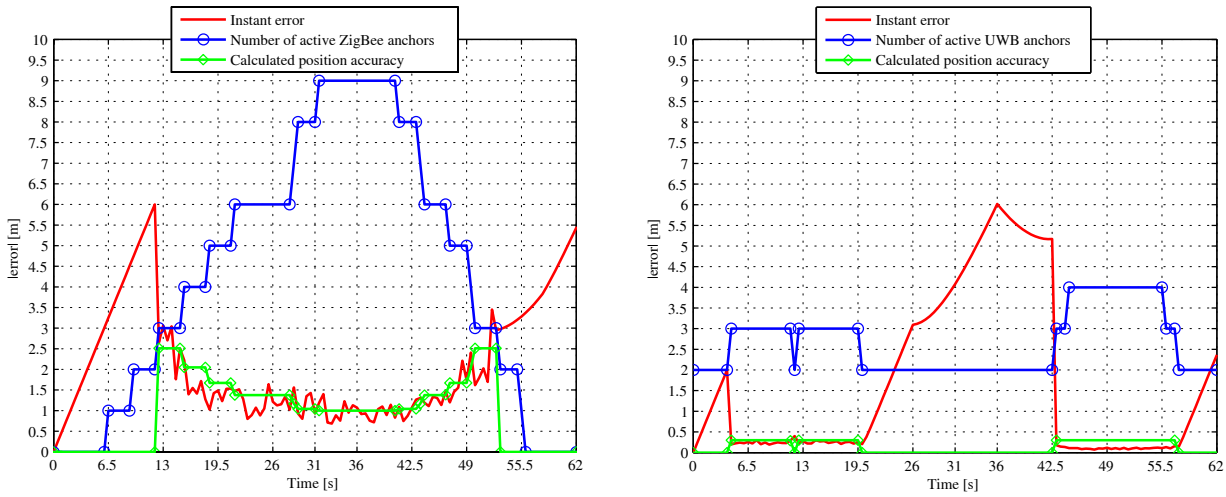


Figure 3.12: Instant error for the ZigBee and UWB networks when used separately

In the subfigure on the right side of Figure 3.12 we show the respective UWB case, when this technology is used alone. In this case the instant error falls to as low as 0.2 m when we have full coverage by the four UWB anchors. However, it lacks coverage during a long period of time, from $t \simeq 19.5\,\text{s}$ until $t \simeq 42.5\,\text{s}$, when the error increases to as much as 6 m.

We can see that when both technologies are used independently we lack coverage during long periods of time, so that the error increases by up to several meters. By using the high-level fusion module of the proposed architecture, we can easily combine both technologies in a single multi-technology *target device* in a flexible manner to obtain fusion positions.

Figure 3.13 shows the instant error obtained when we ask the *Location Server* for fusion positions using the SC, MRC and EGC fusion methods. We see that the three fusion algorithms perform exactly the same when there is zero or only one technology available. This situation happens in the following time intervals: $t \in (0, 12)\,\text{s}$ (none or only UWB), $t \in (19.5, 42.5)\,\text{s}$ (only ZigBee) and $t \in (50, 62)\,\text{s}$ (only UWB or none). The rest of the time, when both technologies are available simultaneously the SC method always select the most accurate technology available in terms of *position.acc* value, while MRC combines both technologies,

assigning them weight depending again on the *position.acc* associated to the estimated position stored in the *Location Server*. Finally, EGC performs rather worse than the other two methods as it does not take into account the *position.acc* value for averaging the available positions. And, as soon as ZigBee positions become much noisier than UWB's, it produces worse fusion position estimations. As result, the proposed MRC and SC fusion methods perform very similarly in this simulation scenario because there is a huge difference in accuracy between ZigBee and UWB technologies. In a real scenario, possibly incorporating different technologies RSS-based, the MRC method would probably perform better, averaging a larger set of noisy data, but it is definitely dependent on the amount of anchors available and the indoor scenario characteristics.

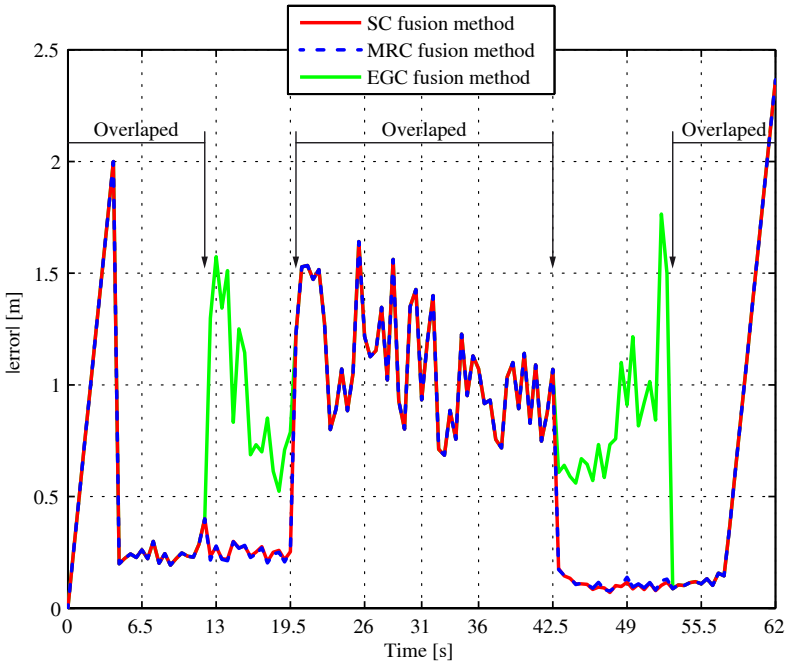


Figure 3.13: Instant error of the fusion positions when using the SC, MRC and EGC methods

The maximum combined instant error obtained with fusion data is less than 2.5 m throughout the whole simulation and is much better than using the WSN technologies separately.

Note that in all cases the fusion positions are pre-calculated in the *Location Server* before being sent back to the *Location Clients*. They are directly packed inside a JSON response message, and the clients only have to display them using an application with a GUI, on a map of the building.

Table 3.2 shows the minimum, mean and maximum size of the messages used during the simulation, in order to evaluate the bandwidth consumption for communicating with the location server. Both the `new_measurement` and `new_position` messages which insert data into the *Location Server* only receive a small OK acknowledge JSON message as a response, which occupies a fixed small amount of data. Other message responses such as the

Message type	Call message size [bytes]			Response message size [bytes]		
	Min.	Mean	Max.	Min.	Mean	Max.
new_measurement (ZigBee)	232	236	238	49	49	49
new_measurement (UWB)	224	229	231	49	49	49
get_measurement (ZigBee)	167	170	171	135	842	1639
get_measurement (UWB)	164	167	168	447	566	785
new_position (ZigBee)	221	225	228	55	55	55
new_position (UWB)	216	219	221	55	55	55
get_position (ZigBee)	219	222	223	159	282	350
get_position (UWB)	216	219	220	156	242	338
get_position (Fusion)	217	221	222	158	311	347

Table 3.2: Message size

`get_measurements` and `get_position` can increase in size depending on the number of measurements or positions requested. In our simulations we always asked for positions in iterations every 0.5 s, for a window size of 0.499 s, so that the number of obtained positions was always zero or one for a specific technology (ZigBee or UWB), and for the fusion positions (zero or only one fusion position is given per call). This is the reason why the `get_position` responses do not grow much in this simulation. However, in the case of measurements, we ask for all the available measurements of the mobile nodes regarding all the anchors of each network, which can be from 1 to 9 measurements per response.

Note that during the simulations we have not used OpenID to authenticate the different clients with the *Location Server*. Therefore, the message size shown in Figure 3.2 shows exactly the size of the JSON messages going through the *Location Server*, not other network traffic involved in the authentication.

3.7 Conclusions

In this chapter we introduce an RTLS architecture supporting multiple technologies. This architecture was designed to provide an easy mean of including new hardware, new algorithms and any kind of LBS by considering client applications of the location estimations of targets of interest.

The architecture also provides a complete API with several methods for interfacing with external modules. Hardware providers can set up a network in the platform based on their own technology transparently, for example, to the algorithm developers. Algorithms can use raw data from one or more available technologies to provide location estimation. And these location estimations are available in real time (or off-line, with access to historical data) for location applications.

The main advantage of the proposed architecture, as opposed to monolithic solutions, is that it allows a generalization and retargeting for new LBS and applications, reducing costs of further development. Other kind of purpose-built systems can be expensive and hard to implement in

practice.

To demonstrate the application of the architecture, we introduced two case studies based on real deployments. And finally, the performance of the platform in a specific scenario has been demonstrated by the validation results.

Chapter 4

Customized Hardware for Localization

In this chapter we describe a developed solution for optimizing costs and improving the diversity of Radio-Frequency Identification (RFID) readers, which are the most costly hardware parts when implementing RFID based Real-Time Location Systems (RTLSs). This chapter is partially based on the publication [133].

RFID is an attractive technology for implementing applications such as localization, detection, security, etc. When deploying these systems indoors, several doubts emerge as to how many readers and antennas are needed in order to minimize costs, but also to avoid tag-to-reader and reader-to-reader interferences. We present a hardware and software solution to overcome this problem. Using a custom designed RF switching box based on state of the art RF switches, we multiply the maximum number of antennas possibly connected to each reader (up to sixteen antennas per reader in our case), increasing diversity. We are thus able to reduce costs of the RFID infrastructure in two ways: by adding RF switches and antennas instead of buying more expensive readers; and building an autonomous reader-plus-switch solution, which does not require any external power supply, only a Power-over-Ethernet (PoE) network connection.

Finally, with an implemented software layer we can remotely schedule and control, in a centralized way, all readers and selected antennas to minimize interferences among readers.

The chapter is organized as follows. In Section 4.2 we introduce the custom-designed RF switching box to gain antenna diversity. In Section 4.3 a software layer designed to communicate, configure and synchronize several RFID nodes is explained. In Section 4.4 we exemplify an indoor location system using RFID, taking advantage of the designed hardware and software parts, and compare the deployment costs compared to a traditional solution. Finally, Section 4.5 is dedicated to the conclusions.

4.1 Introduction

RFID is a means of identifying people or objects via a wireless technology. Nowadays, the number of applications of this technology is increasing due to its flexibility, which goes far beyond the capability of passive and non-unique printed barcodes. RFID tags are sophisticated devices with an Integrated Circuit (IC) that carry field updatable information, making it possible to identify unique objects, and they are not easily counterfeited.

There are different possibilities in RFID technology depending on the specific frequency band and communications protocol used. The protocol of interest to this discussion is the EPC Gen2 (also known as Class 1 Generation 2), standardized as ISO 18000-6C. It is a single worldwide standard working at Ultra-High Frequency (UHF), which defines reader and tag communications operating in the 860 MHz to 960 MHz frequency bands. This range includes the 868 MHz (European) and 915 MHz (American and Australian) ISM bands. It allows inexpensive passive tags to operate without battery and to communicate with the reader at near and far field distances, of up to several meters.

RFID applications usually need to cover wide indoor regions, which require a sophisticated deploy planning using several antennas and readers, and it is always desirable to minimize costs. Commercial RFID readers, the most expensive parts in the system, are usually limited to up to two or four antenna ports, and our solution makes it possible to multiply the number of possible connected antennas, gaining in coverage and antenna diversity.

4.2 Proposed Hardware

The main objective of the designed RF switch board is to increase antenna diversity of RFID readers at optimized costs. We have designed an RF switch board that can be used to obtain eight antenna ports from a 2-port RFID reader, or by combining two boards in tandem, up to sixteen ports from a 4-port RFID reader.

Figure 4.1 shows the two possibilities for connecting one or two RF switching boards to different RFID readers. For this design we have chosen Speedway Revolution RFID readers from Impinj [15], which are industrial-grade RFID readers with two or four independent, bidirectional, and full duplex TX/RX monostatic antenna ports. They also provide a self-powered General-Purpose Input/Output (GPIO) extension port with eight digital opto-isolated control lines (four inputs and four outputs). These lines can be remotely read or switched over Ethernet using standard Low-Level Reader Protocol (LLRP) commands. Moreover, the readers support PoE connections, thereby avoiding the need for external power supplies and so reducing hardware and deployment costs.

Each custom-designed RF switching board (*8:2 switching board* in Figure 4.1) is detailed in Figure 4.2. The main objective was to design an RF switching circuitry able to work with

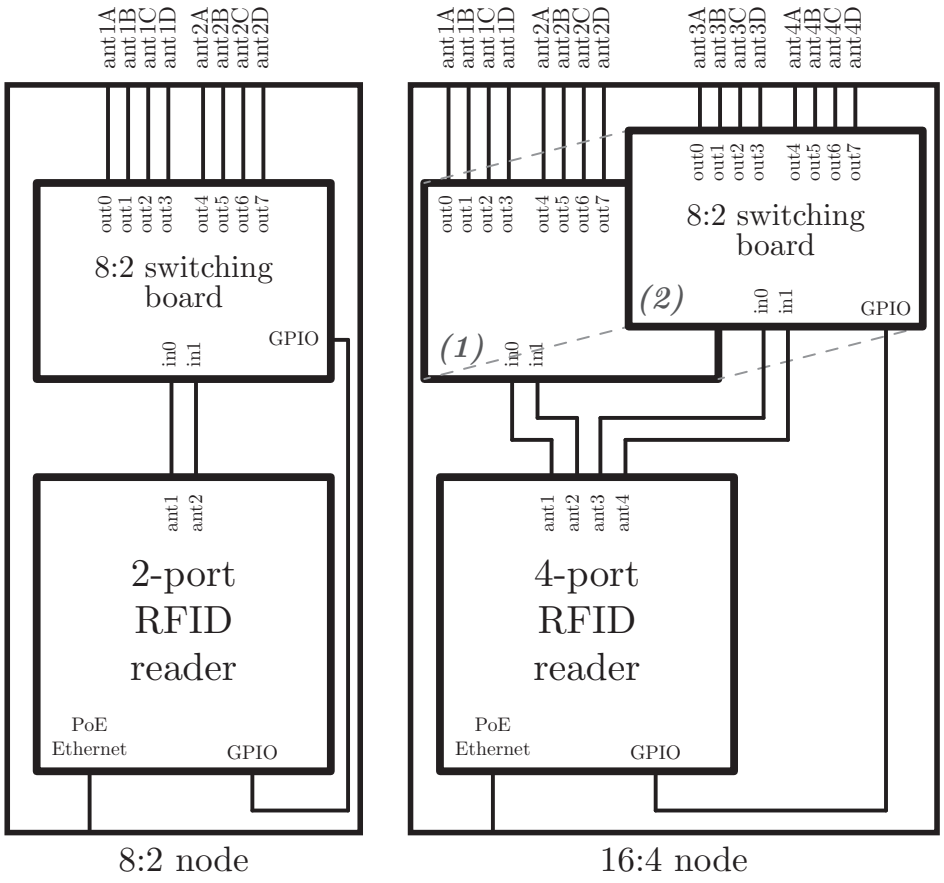


Figure 4.1: RF switching box diagrams

two or four independent input signal of up to 30 dBm (1 W). The inputs of this box would be connected to RFID reader antenna ports working in a monostatic schema at UHF frequency bands, so that we could make different pairs or quartets of ports simultaneously available to the resulting RFID node.

We have used 2:1 RF switch chips, model HMC544E from Hittite Microwave manufacturer [11]. We use multiple switches in a cascade and parallel structure to multiply each port of the chosen RFID reader by a factor of four. The main characteristics of this design are:

- Each RF chip is a 2 W capable SPDT T/R switch able to work from DC to 4 GHz with hot-switching support.

- It has a high input 1 dB compression point (P1dB) of +39 dBm, and a high input third-order intercept point (IP3) of +55 dBm. Note that the maximum input power provided by the Impinj reader is +30 dBm.

- It has a very low insertion loss of 0.2 dB at 1 GHz. As there are two switch chips connected in series per antenna port output, we lose at least 0.4 dB by the switches alone, plus the losses of the SMA connectors and the RF transmission lines on the FR4 PCB.

The operation frequency below 1 GHz allows to chose low cost FR4 PCB material to further minimize costs.

- The isolation between the outputs is 23 dB. This value is sufficient for RFID applications where only fully-passive tags are used. With the maximum input power of +30 dBm in one active port, we have up to +6.4 dBm at a non selected antenna port. This power could be enough to power a passive tag, but only at near-field distances. Therefore, special care should be taken for applications using semi-passive tags even at far-field distances.

Figure 4.2 shows the schematic of the designed board and the truth table of every RF switch.

The connections of the GPIO port pins VCC with V+ and GND with V- are made in order to have TTL levels at the four output pins (*out1* to *out4*). In the reader, these outputs are opto-isolated pulled-up to V+, or connected to V- when they are set to high or low state by software.

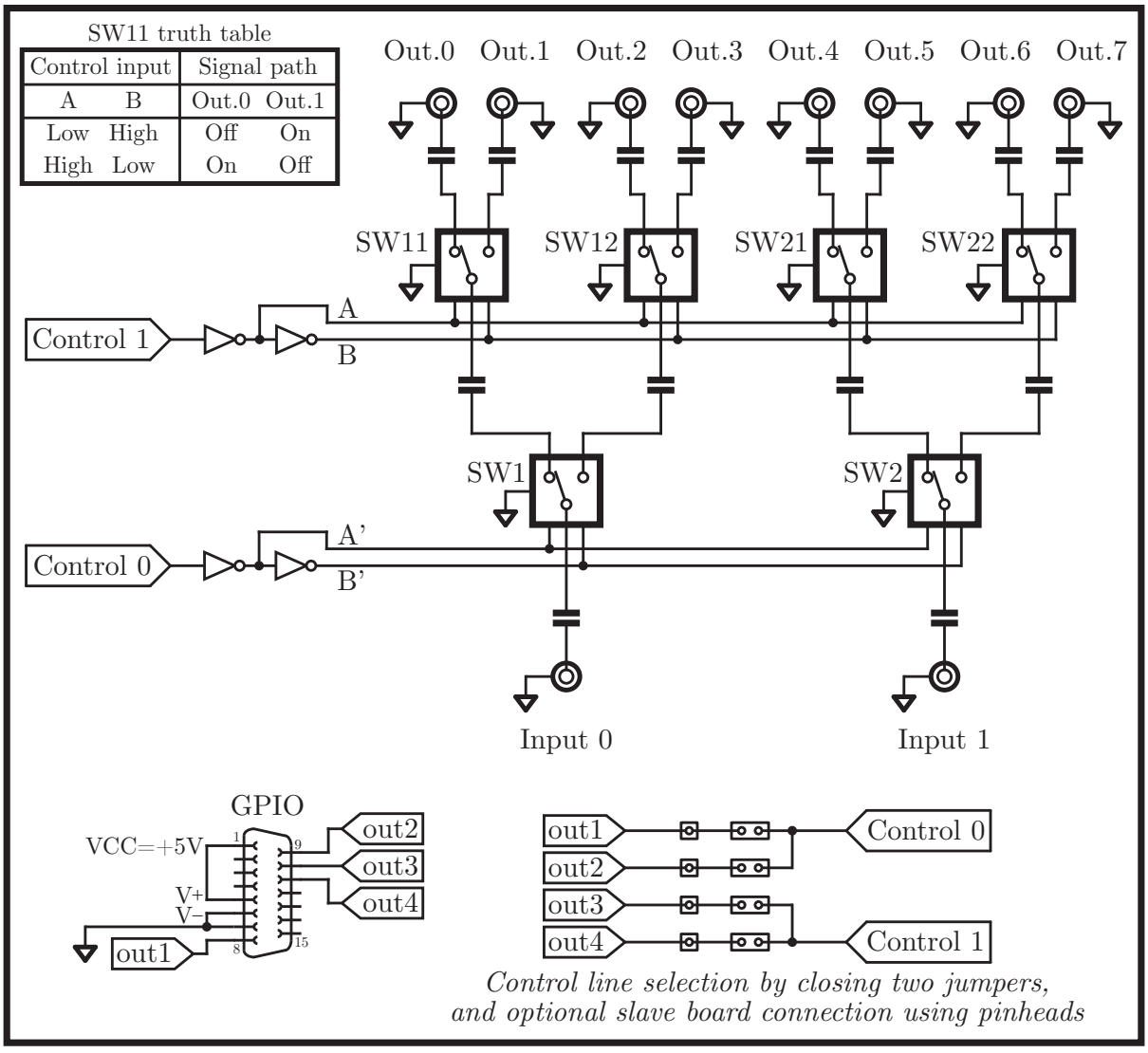


Figure 4.2: 8:2 switching board schematic

Control line		Input							
		0				1			
		Output							
0	1	0	1	2	3	4	5	6	7
0	0	×				×			
0	1		×				×		
1	0			×				×	
1	1				×				×

Table 4.1: Control line truth table of one RF switching board. When using two boards in tandem, inputs 2 and 3 are connected to outputs 8 to 15, and both board switches are independent.

The RF chips are directly powered using their two control lines as shown in Figure 4.2 in the truth table. Using two binary hex inverters we can derive the appropriate VCC&0 or 0&VCC values to the switches control inputs. Therefore, only one control bit per level of RF switches is required, and only two digital control lines to drive the two RF input ports (from the reader) to the eight RF output ports (to the antennas). The first control line *Control 0* is shared among the first level RF switches (*SW1* and *SW2*), while the second control line *Control 1* is shared among the second level switches (*SW11*, *SW12*, ...). Therefore, we only have the four possible states specified in Table 4.1. Four possible port combinations (ports 0&4, 1&5, 2&6 and 3&7) can be selected via the two digital control lines. For the input port *Input 0* one of the output ports *Out. 0* to *Out. 3* can be selected and for the input port *Input 1* one of the ports *Out. 4* to *Out. 7* respectively. This may seem a design flaw but actually in real scenarios we could number the two output ports selected at the same time in a proper manner to solve this issue. This is shown in an example scenario in section 4.4.

We have thus successfully optimized the number of control lines needed from the limited available outputs of the GPIO port. In this way, with the four digital outputs available in the 4-port Impinj RFID readers, we can switch up to sixteen antennas in quartets.

In Figure 4.3 we show the side enclosure views of a 16:4 RF switching box, and in Figure 4.4 two possible front and back panel designs for a chosen extruded aluminium enclosure. We show three photos of the manufactured prototype boards in Figures 4.5, 4.6 and 4.7. Figure 4.8 shows a complete *16:4 node* diagram which includes a 4-port Impinj Speedway Revolution RFID reader, RF adapters and pigtails, cables, etc. attached together inside a metal enclosure.

Finally, Table 4.2 shows a cost estimation of the RF switching boxes, calculated considering one or two 0.04" thick FR4 PCBs, one extruded aluminium enclosure with custom metal faceplates, the Hittite switches and the rest of components by Farnell. The assembly costs are not included so that these are prototype reference costs, not mass production costs. Also note that we minimized the hardware complexity and Bill of Materials (BOM) costs, as no embedded power supply or micro-controller (with firmware development) are needed.

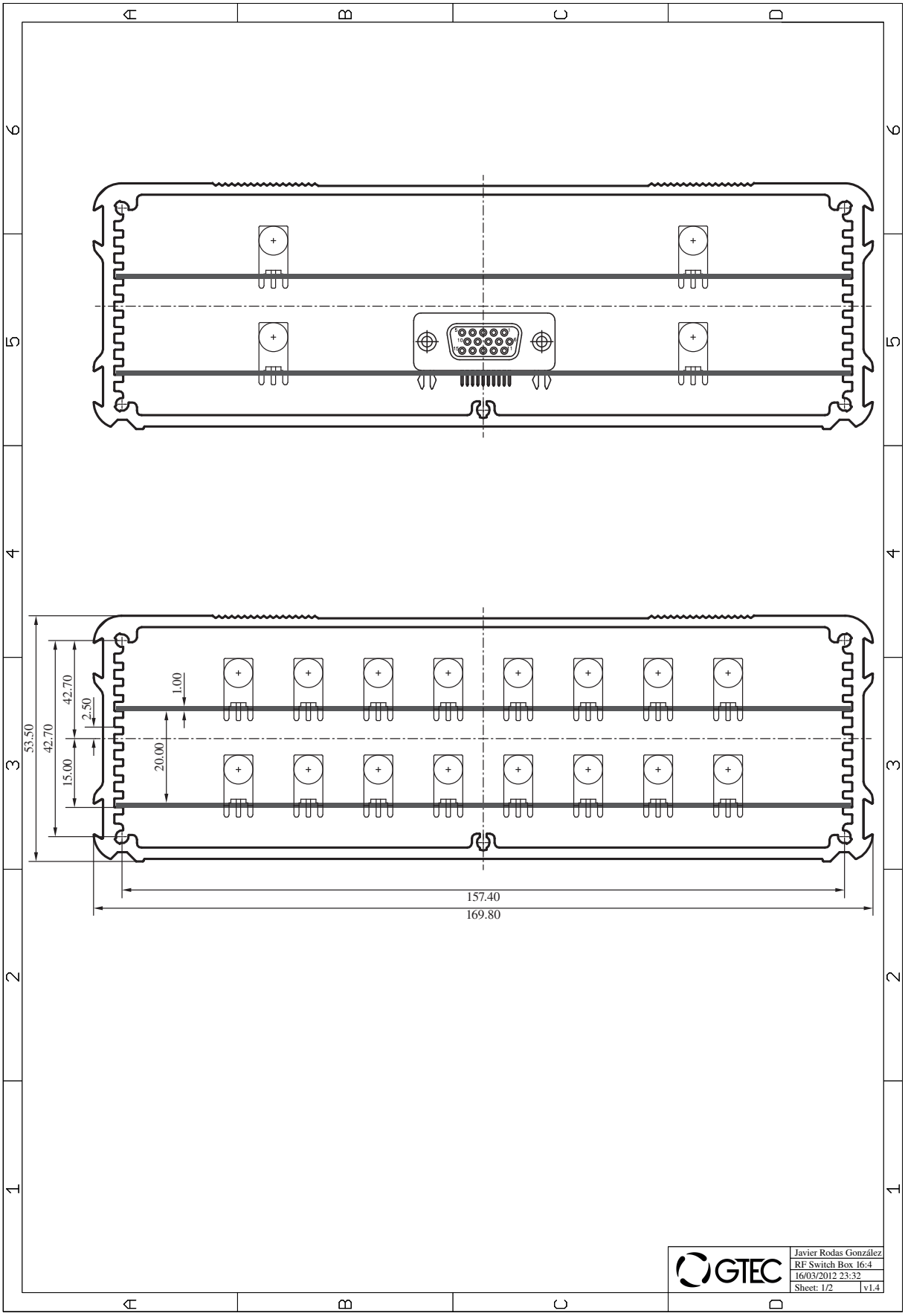


Figure 4.3: 16:4 RF switching box side enclosure view.

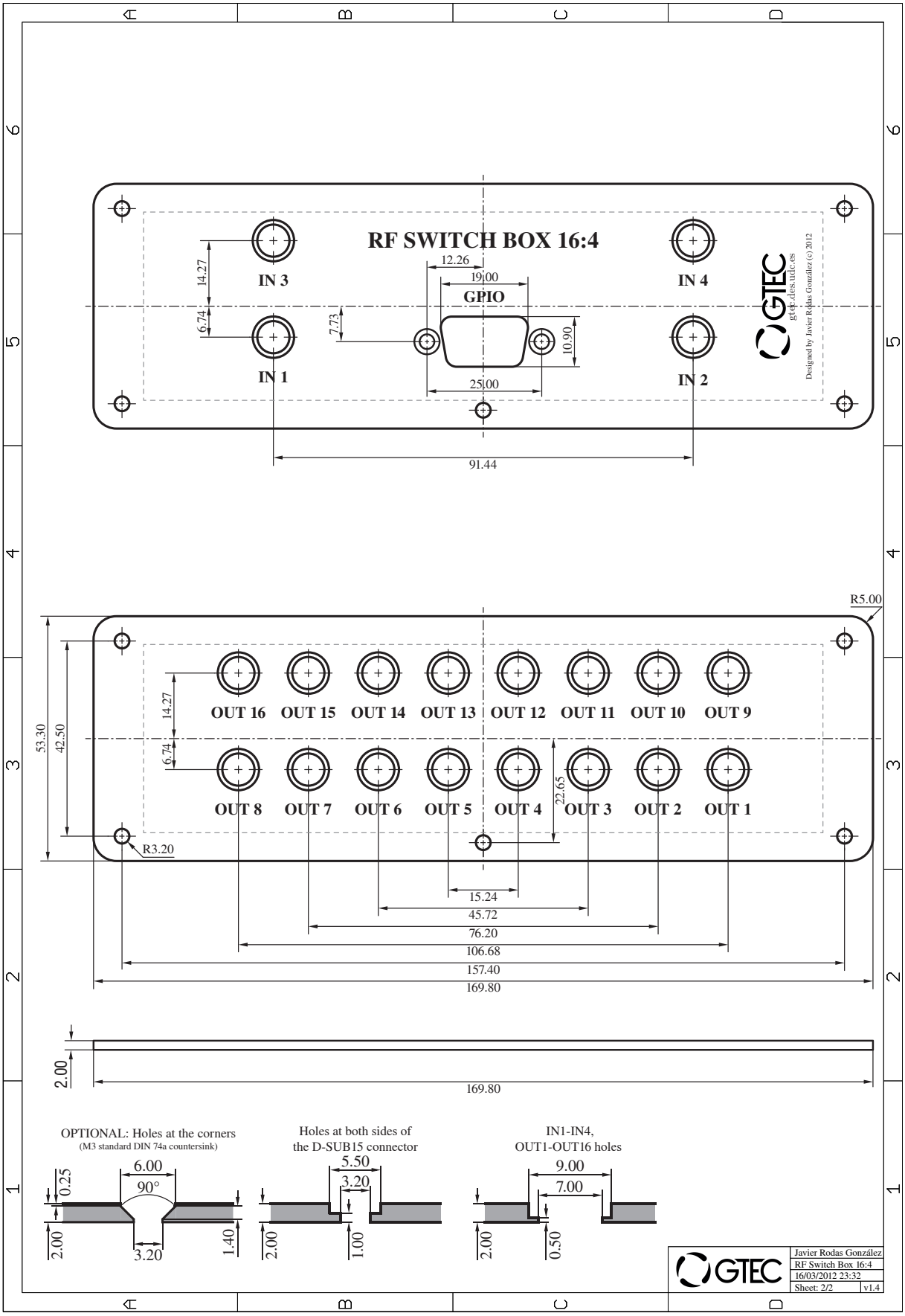


Figure 4.4: 16:4 RF switching box front and back panels.

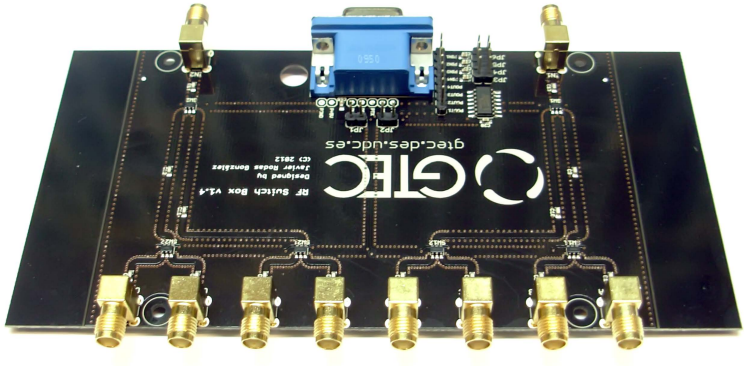


Figure 4.5: *8:2 switching board*.

Figure 4.6: *16:4 switching board* configuration, made up two 8:2 switching boards.

Figure 4.7: *16:4 switching board* configuration inside the extruded aluminium enclosure (without the front and back panels).

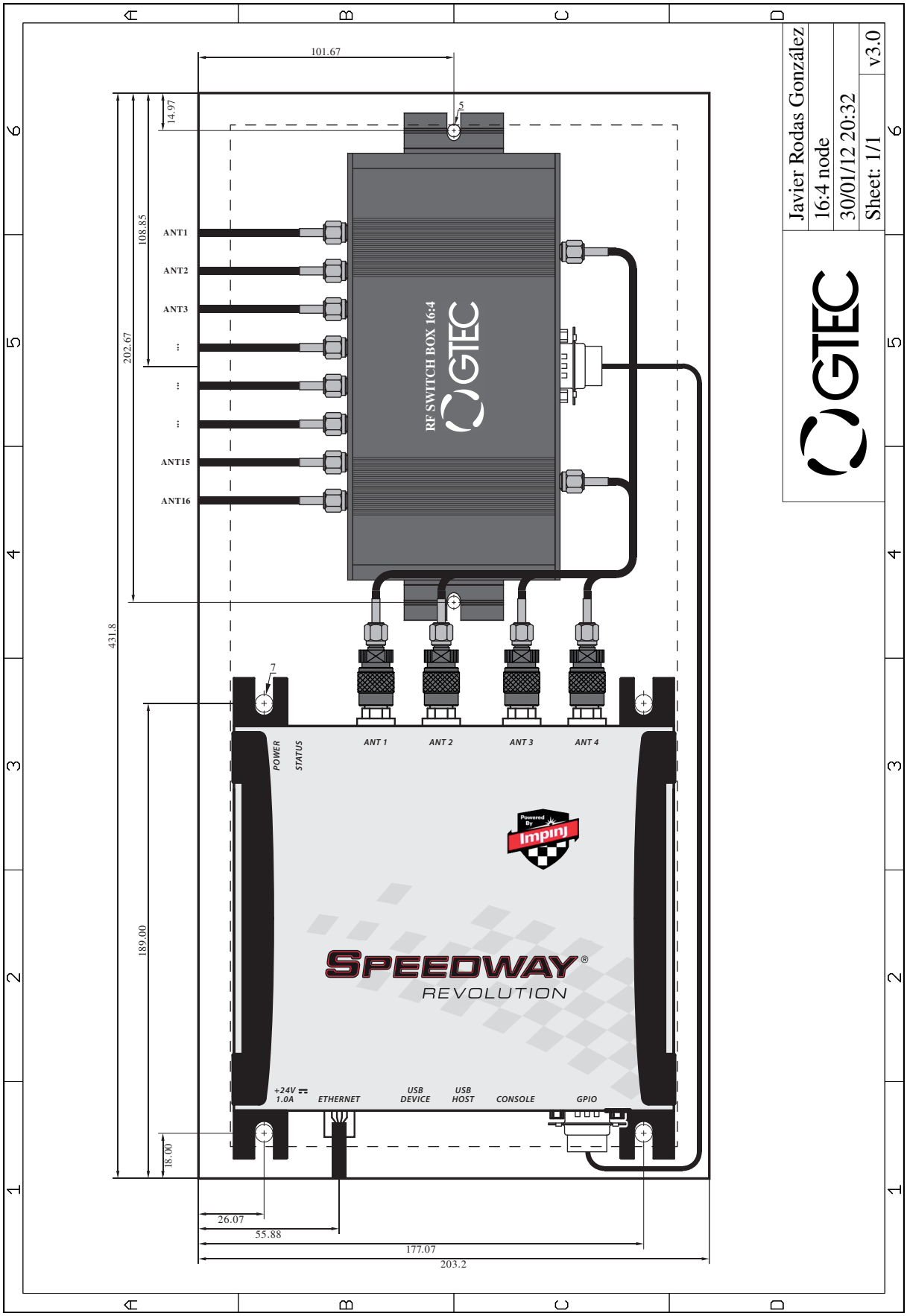


Figure 4.8: *16:4 node* diagram.

Quantity	Price per 8:2 switching box	Price per 16:4 switching box
10	77,07€	104,11€
50	53,51€	72,74€
100	45,58€	69,03€

Table 4.2: Approximate price per RF switch box.

4.3 Proposed Software

We have implemented a software layer able to abstract the communication, configuration and time synchronization of several RFID nodes (composed of a reader and an RF switch board) which should work combined in the same scenario. From a centralized host we can remotely configure the transmit power of each output antenna (to control the effective transmit power depending on the cable length of each segment), the active ports of the readers, and the selected antennas by using the RF switching boards (through the GPIO port of each reader).

Figure 4.9 shows the typical duty cycle of an RFID system using multiple output antennas with one or two added RF switching boards.

- Within the time period *measurement window*, each reader is started by an Ethernet command and continuously sends query commands and processes responses from the tags. Each reader continuously senses the scenario using its two or four available integrated ports by cyclically switching among them. For example, in this step, we would extract Received Signal Strength (RSS) information from tag responses in a tracking application.
- After a predefined period of time each reader automatically stops.
- Then, the antenna switching at the RF switching unit is performed, during the short time

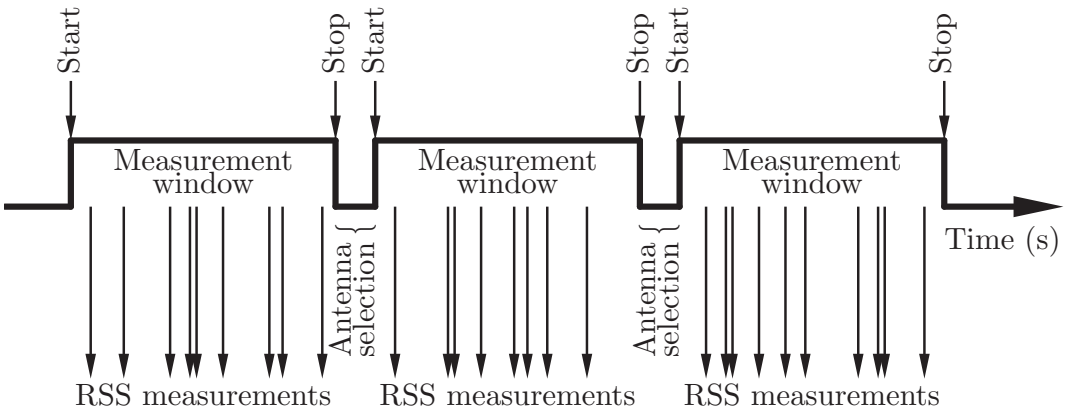


Figure 4.9: RFID proposed software duty cycle diagram.

period *antenna selection*. Depending on the reader chosen, with two or four antenna ports we can scan more areas with a specific switching configuration during the same *measurement window*. But it is necessary to schedule the antenna switching appropriately to cover the whole scenario with a few steps, and without interfering with other readers using equivalent switching schemas. In section 4.4 there is an example case study of a tracking application which is covered using this strategy.

Note that without external triggering hardware the time synchronization can not be guaranteed in terms of a few milliseconds due to unpredictable Ethernet network latency [81], even in the case of a dedicated network infrastructure. However, when the RFID application does not require such precise time resolution (i.e. when tracking walking people) we can simply sequentially start each reader by sending a network command. We can make the reader work for a predefined period so that it stops automatically without any additional network command.

Typically, the *measurement window* will be much longer than the *antenna selection* period, so that the performance impact caused by the RFID sensor network inactivity should be insignificant. For applications demanding a better time synchronization the Impinj readers can be started by an external trigger by pulling or rising one of the control line inputs available at their GPIO port. Extra hardware synchronization units should be used in this case. The synchronization unit designed at the Vienna University of Technology [81] could be valid for this purpose.

4.4 Example Case Study: An Indoor Location System

As a case study for the previously designed hardware and software parts, in this section we will explain how to deploy and configure an indoor localization system with them. Finally, we will compare the costs of our solution compared to the case of not using any RF switching box.

In many indoor scenarios it is desirable to estimate at least the position (optionally the speed and direction) of a moving person, animal or object (in general any target tag). In our example we restrict ourselves to only passive tags to avoid undesired tag activations by non-active antennas.

For localization we can use linear or circular polarized antennas. In general the latter are preferable to avoid non detections due to the orientation of the tags. There are also different passive tag antenna designs, as for instance the Impinj True3D omni-directional technology [14] that can maximize signal reception at more tag orientations, as an alternative to traditional dipole antenna technology.

We have carried out some measurements in an indoor scenario for distances and angles at the points shown in Figure 4.10. The resulting mean RSS values obtained for two different kind of passive tags are shown in Figure 4.11. The purpose of this test was to measure the effective side

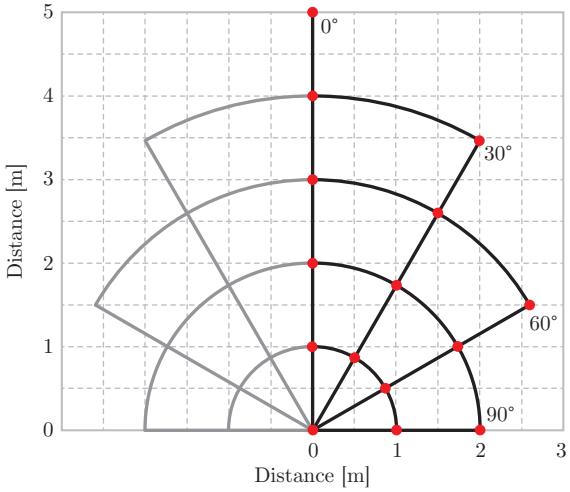


Figure 4.10: Measurement points in an indoor sample scenario

coverage of typical RFID antennas with different real passive tags, when using several transmit levels. This allows us to discover hardware limitations and determine the minimum antenna separation that should be chosen to properly cover an indoor scenario minimizing blind areas, as well as the maximum cable length that should be deployed, and therefore the introduced loss.

The antenna used was a far-field Laird Technologies S8658-PR right-hand circular-polarization antenna [20], with 3.85 dBd gain. It was mounted at 1 m height above the floor. The two tags were a Frog 3D [33] and a DogBone [32] passive tags, which were mounted on the back of a plastic chair, separated 15 cm from each other and orientated vertically. The chair was moved towards the points depicted in Figure 4.10 always maintaining the same orientation.

Figure 4.11 shows the mean RSS values obtained when transmitting at different powers (20 s at each power level, obtaining typically fourteen measurements from each tag in the favourable cases). Looking at the results it is clear that *tag2* behaves better for lower transmit power and

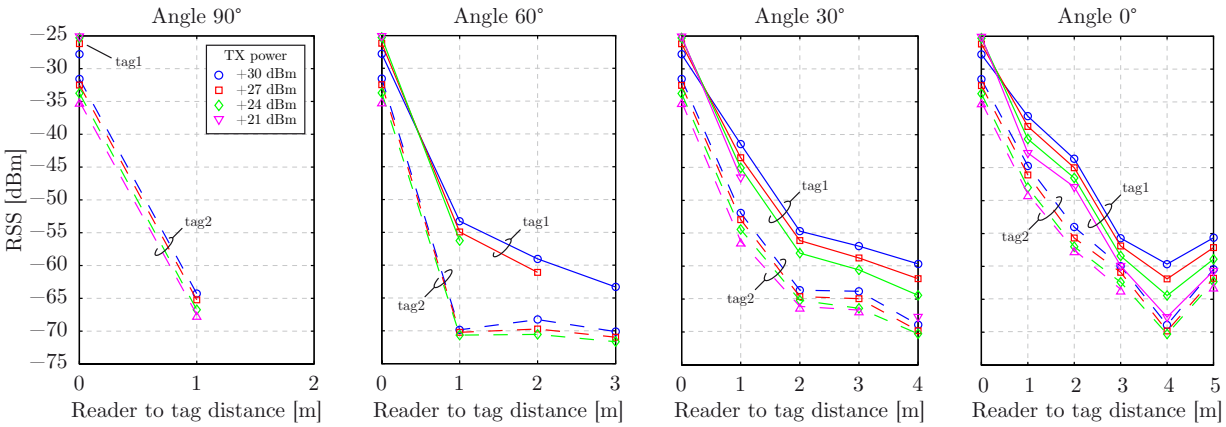


Figure 4.11: Mean RSS experimental values at four different angles between the antenna and the tag, repeated at different linear distances among them.

is thus more sensitive than *tag1*. However, the RSS levels of *tag1* are always greater, when measurements are obtained.

Therefore, to guarantee a side coverage in an indoor scenario we could separate this kind of antenna not farther than 2 m from each other along the walls. For applications demanding more precise localization we could reduce the distance between antennas or use antennas with a wider horizontal coverage.

In Figure 4.12 we show a purely hypothetical example scenario. It could be a possible scenario for indoor localization with three corridors where people or other object positions could be estimated, along the corridor and from side to side. In the example the corridor on top demands a higher accuracy so we deploy more antennas with reduced spacing between them. In other cases we use antennas with a wider horizontal propagation pattern and separate them more, as those corridors require less precision. The three nodes on top are 16:4 nodes, as we desire to read from four different sector areas within the same measurement window, while the lower ones only process two active sectors at each time window. For this last case we could just use a 4-port RFID reader, but it is even cheaper to buy a 2-port reader plus a RF switching

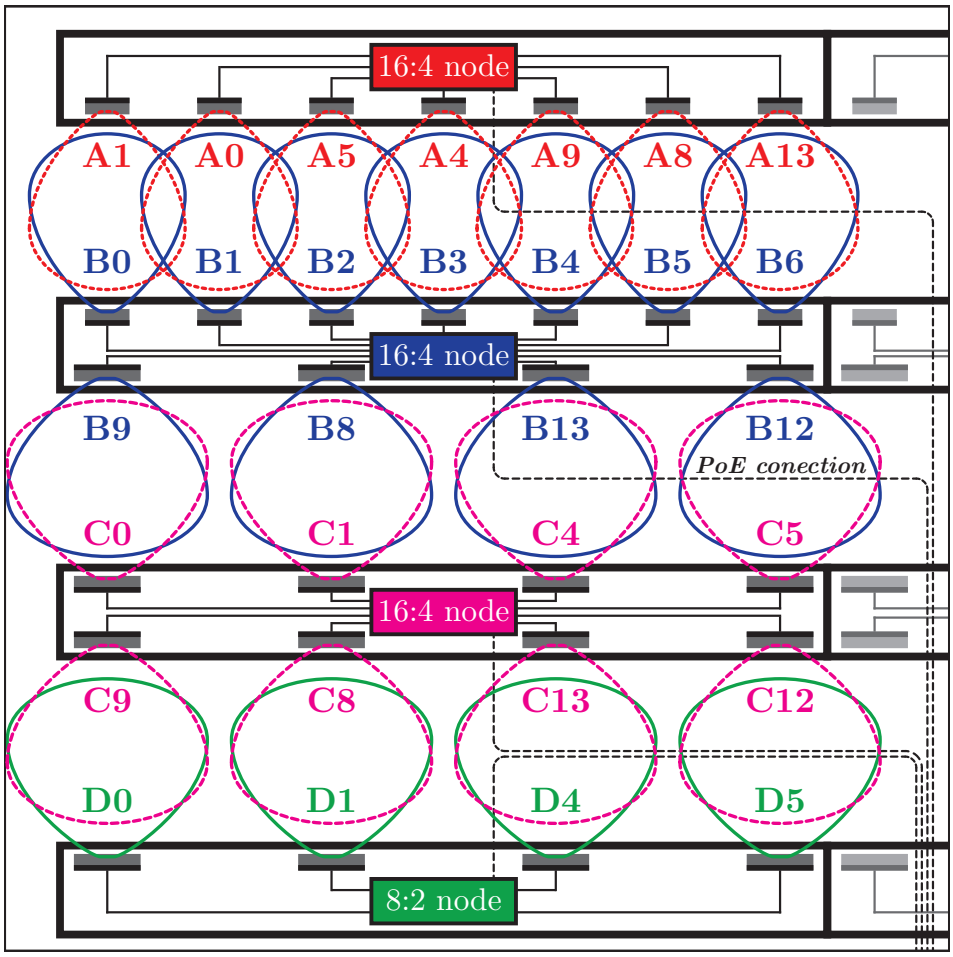


Figure 4.12: Deployment diagram

box than a 4-port reader. In all cases we did not really need to use the maximum number of supported antennas (up to eight or sixteen depending on the node), but in all cases we gain antenna diversity by using our proposed hardware solution, and we optimize the costs as the most expensive parts are the RFID readers.

To maximize the covered area of the indoor scenario at each stage of the measuring algorithm we have to choose the appropriate number of antennas connected to each output of our switching boxes, remembering that our design has the restriction of selected output in fixed pairs as it was shown in Table 4.1. The chosen antennas must guarantee that the same region is not covered by two antennas from different readers at the same time, to avoid reader-to-reader interferences. One possible numeration for solving this example scenario is shown in Figure 4.12, the four steps detailed in Table 4.1 and schematically shown in Figure 4.5. The corresponding control bits are shown in Table 4.3. Depending on the cable length to each antenna we also would have to estimate the optimum transmit power. It should probably be set differently for each output port at each step of the algorithm.

Step	Selected antennas				Control bits			
	A	B	C	D	A	B	C	D
0	0 & 4 8 & 12	0 & 4 8 & 12	0 & 4 8 & 12	0 & 4	00 00	00 00	00 00	00
1	1 & 5 9 & 13	1 & 5 9 & 13	1 & 5 9 & 13	1 & 5	01 01	01 01	01 01	01
2	0 & 4 8 & 12	2 & 6 8 & 12	0 & 4 8 & 12	0 & 4	00 00	10 00	00 00	00
3	1 & 5 9 & 13	3 & 7 9 & 13	1 & 5 9 & 13	1 & 5	01 01	11 01	01 01	01

Table 4.3: Cyclic switching steps table for the proposed localization scenario example.

Finally, Table 4.4 shows the different costs needed to implement the example localization infrastructure shown in Figure 4.12, with or without our proposed RFID nodes. Note that the costs of antennas and cables are not included in the cost table, as they are the same in both situations.

		Our solution		**Traditionally**	
Description	**Unit price**	**Qty**	**Price**	**Qty**	**Price**
4-port RFID reader	1144,90	3	3434,70	8	9159,20
2-port RFID reader	1000,40	1	1000,40	0	
16:4 switching box	104,11	3	312,33	0	
8:2 switching box	77,07	1	77,07	0	
TOTAL PRICE (€)			4824,50		9159,20

Table 4.4: Total hardware cost estimation with or without considering our RF switch box solution.

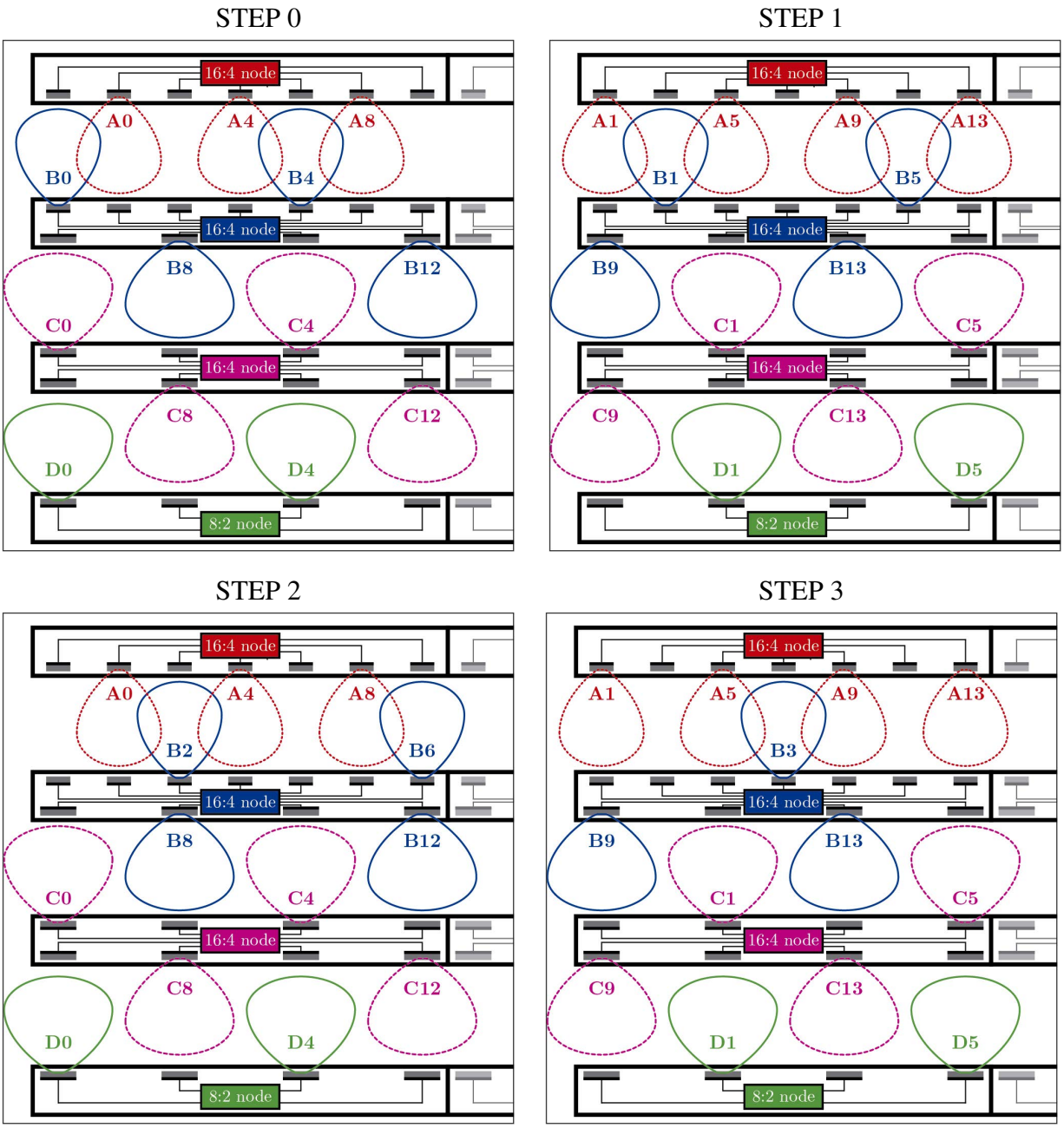


Table 4.5: Cyclic switching steps for the proposed example scenario, represented graphically.

4.5 Conclusions

We have introduced a hardware and software solution to improve the diversity of RFID readers based on ISO 18000-6C EPC Gen2 standard. The solution introduced presents a good trade-off between hardware and software complexity while minimizing costs. The proposed RF switches multiply the number of antennas available in RFID readers by a factor of four, and they are controlled using their GPIO port. The cost comparison results demonstrate that our solution is cost-effective.

Chapter 5

Design of Antennas for Localization

Antenna design is a very important research field for improving the performance of Wireless Local Area Network (WLAN) communications but also of Wireless Sensor Network (WSN) communication systems, mainly used for ranging purposes in Real-Time Location Systems (RTLSs). In this chapter we explain some research work related to this field, partially based on the publications [122, 132]. The rest of the chapter is organized as follows. In section 5.1 we provide an introduction to the Electromagnetically Coupled Patch (ECP) antenna design problem and objectives. In Section 5.2 we present a first numerical model design for a one-patch ECP antenna, which is the base for the other two array antennas presented in Sections 5.3 and 5.4. In Section 5.5 we show how real measurements were obtained in an anechoic chamber to validate the antenna prototypes (built for the first and third antenna numerical models).

5.1 Introduction

The antennas used for both WLANs (for communication among computers), and WSNs (for ranging purposes in RTLS) should radiate properly, with a wide coverage, minimizing reflections due to the walls and floor in order to maximize the radiation in the area of the receiver equipment.

For this kind of application, it is convenient that the radiation pattern follows certain requirements:

- High gain.
- An angular coverage that is area as wide as possible (i.e. $|\theta| > 45°$).
- Good bandwidth, in terms of input impedance and radiation pattern integrity.
- Linear polarization.

- Small size, low cost and easy to manufacture, which are desirable for portable devices in general.

ECPs have recently caught the attention of antenna designers because of their absolute high gain values and their excellent broadband input impedance [76]. It can be pointed out that ECPs have much better bandwidth than antenna models designed with single patches, which are embedded on one-layered substrates which usually provide an impedance bandwidth below 5% (usually near 2% [73]). Their radiation pattern has vertical linear polarization, a convenient way to increase the efficiency of the radiation transmission. The later makes the antenna suitable for both indoor and outdoor applications, such as for WLAN [101], Bluetooth [136], WiMAX [118], or any other technology, either for massive data transmission or for localization purposes. Unfortunately, most of the ECP models found in the literature have linear polarization along the θ unit vector, considering that the antenna plane matches the x-y plane of its global Cartesian coordinate system (see, for example, [76]). This feature could reduce the polarization efficiency because the electromagnetic signals coming from several transmit or receive antennas of the same type are difficult to align to obtain a good polarization coupling.

We continued the research work of J. C. Brégains, who had contributed with several ECP results [77] and ECP array designs [76], in particular the case where multiple patches were aligned along circular paths. In the following sections we show three antenna designs embodying the same ECP array grouping philosophy, but using linear arrangements for improving the linear polarizability character of the radiated field. For two of them, we built real antenna prototypes and compared their measurements (performed in an anechoic chamber) versus simulation results.

For the numerical models of each of the three antennas presented, we have defined the following performance objectives, having taken into account the spherical coordinate system and angle naming shown in Figure 5.1 (left), and how the built antenna prototypes would be mounted on an anechoic chamber positioner (able to performing azimut and roll rotations as shown in Figure 5.1 (right)):

- Optimize the $|S_{11}|$ vs. frequency curve to keep it under $-10\,\mathrm{dB}$ within the frequency range bounded by f_L and f_H (lower and upper frequencies)[1]. The frequency range $BW_{\%}$ and the central frequency f_C are computed as follows:

$$BW_{\%} = 100\,\frac{f_H - f_L}{f_C}, \quad \text{with } f_C = \frac{f_H + f_L}{2}. \tag{5.1}$$

[1] In general, the cutting frequencies f_L and f_H are defined as the minimum and maximum frequency values, respectively, within which a given parameter fulfills the requirements of the designer. In this case, it was they specify the range (frequency band) within the $|S_{11}|_{\mathrm{dB}}$ is below -10.

- Maximize the minimum angular coverage zone $\Delta\theta_{\min}$ on the $\varphi = 0°$ and $\varphi = 90°$ planes within which the minimum gain G is above 1 dBi[2] [76]. The gain is expressed in dBi as

$$G_{\text{dBi}} = 10\ \log_{10}(G)\ [\text{dBi}] \tag{5.2}$$

- Maximize the minimum axial ratio, which is calculated as the relationship between the amplitudes of the vertical and horizontal electric fields

$$\rho_{\min} = \min\left[20\ \log_{10}\left(\frac{|\mathbf{E}_y|}{|\mathbf{E}_x|}\right)\right] \tag{5.3}$$

within both the coverage region and frequencies greater than f_L and smaller than f_H, thus guaranteeing a good linear polarization parallel to the y axis (see the Figure 5.1 (b)).

These objectives are pursued during each antenna design stage, with the help of a Finite Difference Time Domain (FDTD) software simulation environment.

At the time of designing each antenna, different frequency bands were considered, particularly to fulfill different WiFi and WiMAX requirements, but the numerical design techniques explained for all the antennas can be straightforwardly adapted to all kind of frequency ranges and, in principle, could be valid for several indoor or outdoor WLANs and WSNs.

[2]The decision to choose 1 dBi is, somehow, arbitrary. This is the typical minimum value specified in general, but it could be 3 dBi, which is an even better figure for indoor receive antennas. Moreover, the minimum gain is not the only important factor, but also the gain outside the coverage area of the antenna, which should be low in comparison with the main lobe antenna gain.

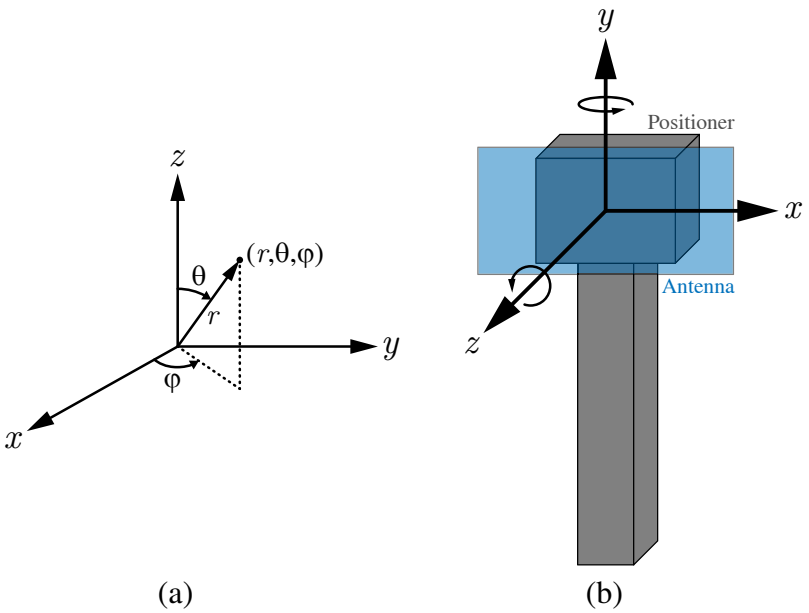


Figure 5.1: (a) Spherical coordinate system. (b) Positioner system representation, where an antenna is mounted in front of it, able to perform azimut (Y axis) and roll (Z axis) turns.

The numerical models shown in the next sections are particularly constrained to the following frequency bands:

- One-Patch antenna: WiFi $5.47 - 5.65$ GHz band [34].
- Three-Patch antenna array: WiMAX $3.4 - 3.6$ GHz band [35].
- Five-Patch antenna array: WiFi, WiMAX $5.0 - 6.0$ GHz band.

5.2 One-Patch Antenna Design

In this section we present a numerical model of a simple and rectangular ECP antenna composed of one radiating patch, prepared for operating on the $5.47 - 5.65$ GHz WiFi band [34]. This is the base for the design of the subsequent array antennas presented in Sections 5.3 and 5.4.

5.2.1 Numerical Model Design

A simulation tool based on the FDTD method [30] was used to design the antenna model, which is composed of a rectangular *RP* radiating patch as shown in Figure 5.2. The l_{RP} and w_{RP} values represent the length and width, respectively, of the *RP* patch. Such an *RP* is electromagnetically coupled to a *SL* microstrip line, of size $l_{SL} \times w_{SL}$, the said microstrip being fed by a single 1 mm diameter pin (of the RF connector shown in the figure).

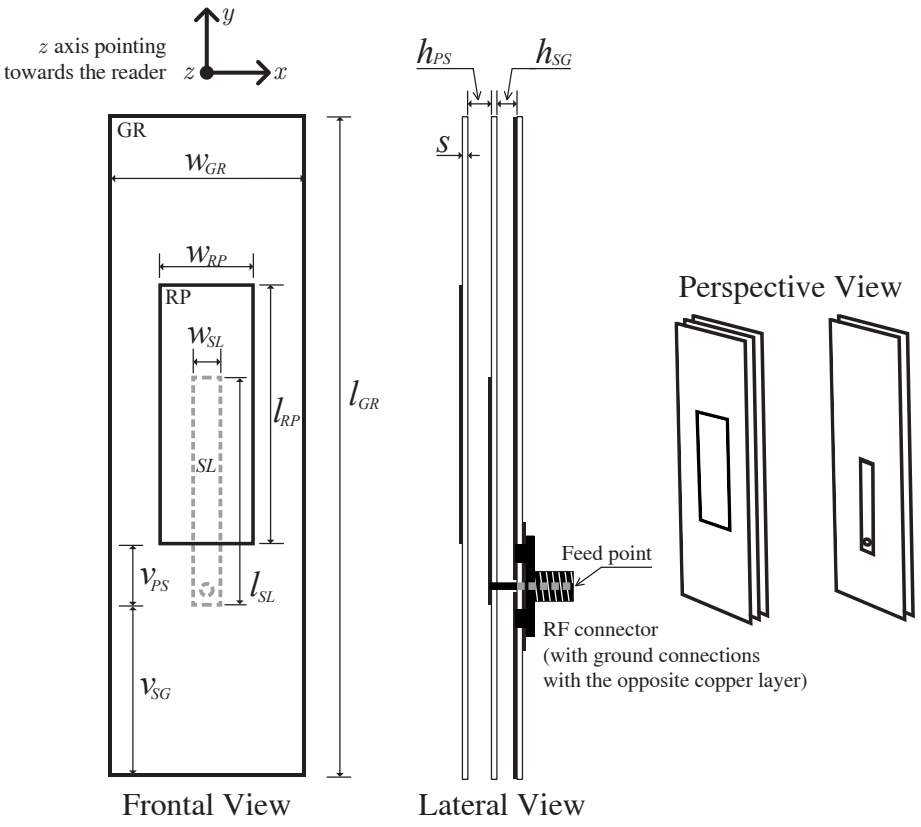


Figure 5.2: Front, side and perspective views of the one-patch antenna geometry.

Both *RP* and *SL* planes are parallel to the *GR* ground plane, and each of them is supported by a thin Printed Circuit Board (PCB) substrate.

5.2.2 Design Technique

To fulfill the specified objectives given in Section 5.1 we follow the two steps described in the Figure 5.3 flow-chart.

We start by fixing the working central frequency, which in turn fixes the working wavelength, and next, based mainly on such a wavelength, we set the initial dimensions of both *RP* and *SL*. Next, we enter a trial and error loop simulating the $|S_{11}|_{dB}$ vs. frequency curve around the required band and testing if the bandwidth and frequency range cover the desired prerequisites. If the bandwidth does not reach a minimum, then we modify the separations between layers (h_{PS} and h_{SG}) and the microstrip line dimensions (w_{SL} and l_{SL}), and we simulate again. Next, if the central frequency does not fit the requirements we then change the patch dimensions (w_{RP} and l_{RP}), and simulate again. This process is repeated as many times as needed until the specifications are fulfilled.

We apply 5% changes in general to the h_{PS}, h_{SG}, w_{SL}, l_{SL}, w_{RP} or l_{RP} for the two trial-and-error steps, and we reduce them to 1% as soon as we approach the goal objectives.

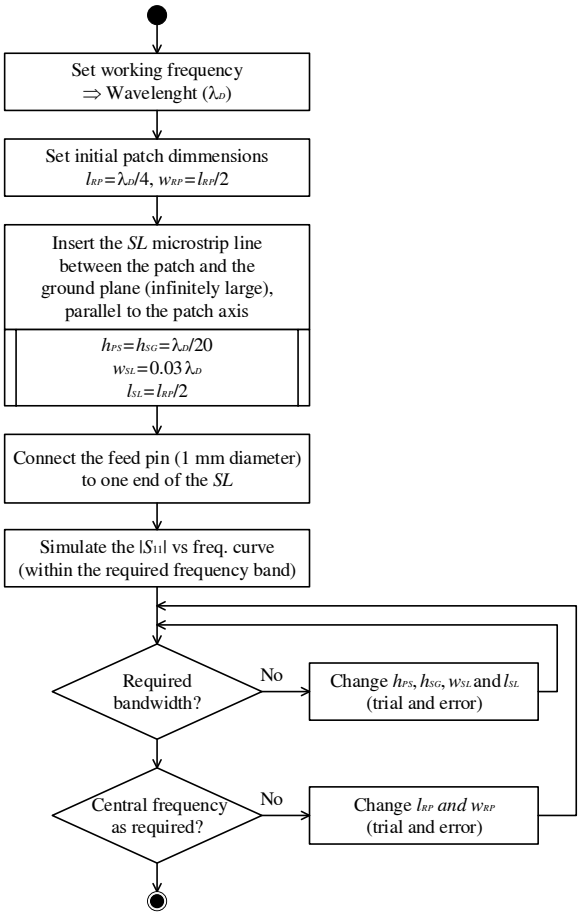


Figure 5.3: Trial and error design technique for the one-patch antenna model.

5.2.3 Results

Table 5.1 summarizes the optimized geometrical parameters according to the aforementioned requirements, as well as some design parameters.

w_{GR} [mm]	l_{GR} [mm]	w_{RP} [mm]	l_{RP} [mm]	w_{SL} [mm]	l_{SL} [mm]	v_{PS} [mm]	v_{SG} [mm]	s [mm]	h_{PS} [mm]	h_{SG} [mm]
27.51	60.01	6.50	19.53	3.94	19.15	5.54	17.47	0.50	2.50	3.50

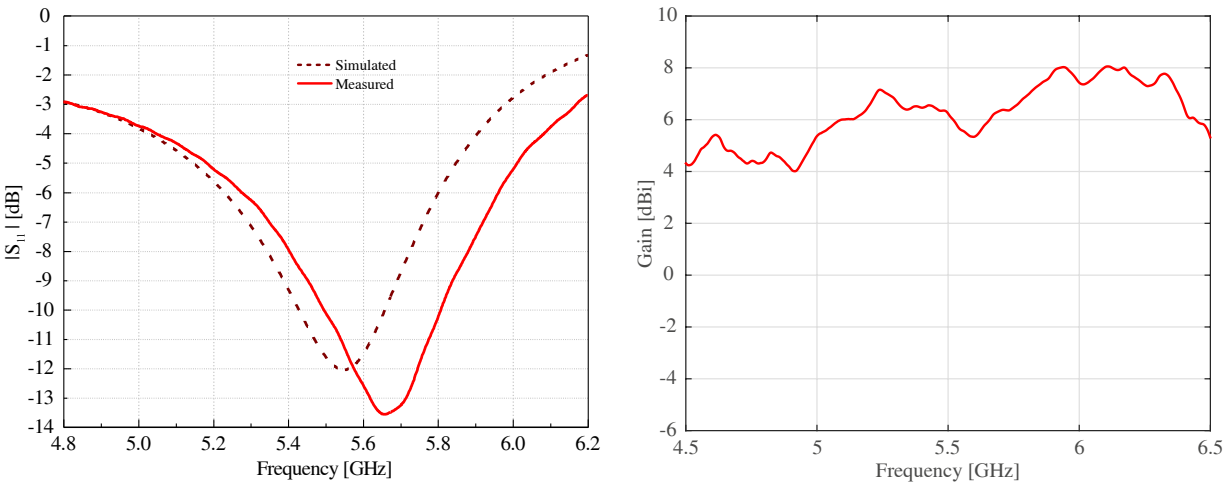
f_L [GHz]	f_C [GHz]	f_H [GHz]	$BW_{\%}$	$G_{\max}$ [dBi]	$\rho_{\min}$ [dB]	$\Delta\theta_{\min}$ ($\varphi = 0^o$) [degree]	$\Delta\theta_{\min}$ ($\varphi = 90^o$) [degree]
5.425	5.541	5.657	4.19	8.312 at f_C	> 18.00	162	68
(5.495)	(5.65)	(5.805)	(5.49)	(6.97 at f'_H)	(> 11.93)	(136)	(69)

Table 5.1: One patch antenna parameters. For the meaning of the symbols, see text and Figure 5.2. The values between parentheses indicate measured parameters.

The largest antenna dimension is 6 cm. It covers a wide frequency band of 4.19% from $f_L = 5.425$ to $f_H = 5.657$ GHz, with a maximum gain $G_{\max} = 8.312$ dBi at f_C.

The antenna fulfills the design parameters over the WiFi band ($5.47 - 5.65$ GHz) [34].

In the Figure 5.4 (a) we show the $|S_{11}|_{dB}$ curves obtained with the FDTD simulation tool and with the corresponding measurements. The measurements are slightly shifted towards larger frequencies, probably due to the SEMCAD tool computation errors, combined with the estimation accuracy of the experimental value of ε_r, which probably differs from the specified in the simulations (taken from the Rogers laminate datasheet, specified at 10 GHz [28]). In Figure 5.4 (b) we show the measured copolar gain $G_0 = G(0°, 0°)$ versus frequency curve.



(a) (b)

Figure 5.4: (a) $|S_{11}|_{\text{dB}}$ reflection coefficient curves. (b) Measured copolar gain $G_0 = G(0°, 0°)$, in dBi.

Figures 5.5 and 5.6 show, respectively, three different views of the experimental prototype. The antenna was constructed with 3 PCBs with RO4000-type dielectric substrates from Rogers Corp. [28], all of them having thickness $s = 0.5$ mm, with relative permittivity $\varepsilon_r = 3.38 \pm 0.05$ and $\tan\delta = 0.0027$ (typical values at 10 GHz and 23°C [28]). A SubMiniature version A (SMA) connector has been used to connect the antenna to the matched (50 Ω) transmission line. The dielectrics have been fixed through the use of M3 nylon 6–6 screws, obtaining the separation distances h_{PS} and h_{SG} by means of 0.5 mm thick M3 metal washers. Both the numerical and the simulation results show the seemingly counterintuitive result that the use of such metal washers does not produce any significant influence either on the field measurements or on the reflection coefficient $|S_{11}|$ of the antenna. Nevertheless, this is easily understandable since those metal components are below the influence zone of the radiating patch.

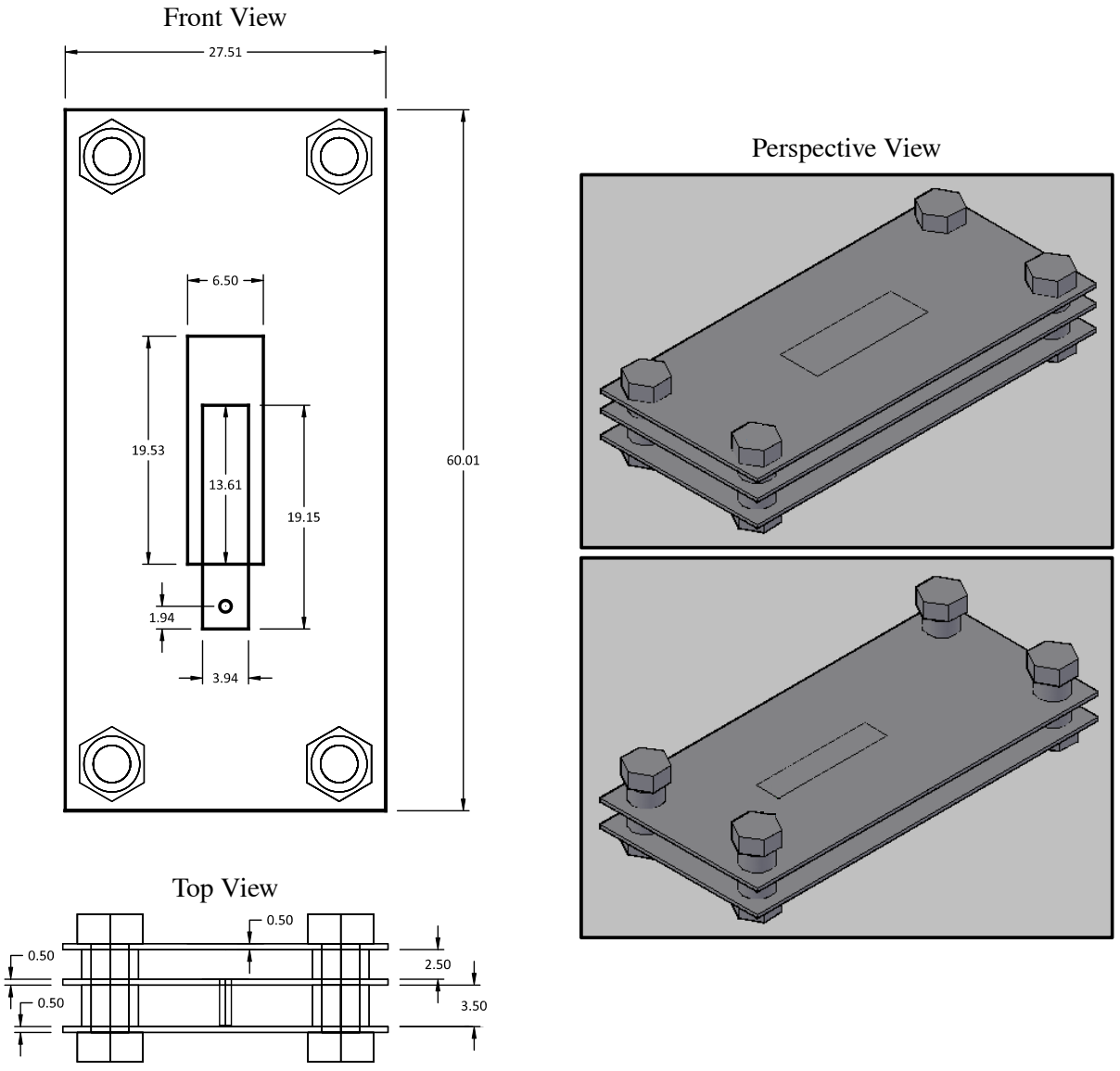


Figure 5.5: Numeric model 3D representation of the one-patch simulated antenna.

Figure 5.6: Pictures of the one-patch experimental prototype.

Figure 5.7 represents the 3D normalized copolar gain pattern, in dB, simulated at the central frequency of 5.541 GHz (f_C for the numerical model), showing an excellent angular coverage over the $\varphi = 0°$ (horizontal) as well as over the $\varphi = 90°$ (vertical) planes, as specified in the design requirements.

In the left hand side of Figure 5.8 we show the absolute horizontal gain pattern ($\varphi = 0°$, $0 \leq \theta < 360°$) at the lowest f_L, central f_C and highest f_H frequencies. In this frequency range we observe a stable pattern (with low ripple), for the whole coverage zone, and nearly the same pattern along the different frequencies.

The minimum horizontal coverage zone $\Delta\theta_{\min} = 162°$ (from $\theta = -81°$ to $\theta = 81°$ the absolute gain is over 1 dBi) is obtained at the f_L frequency. The measured minimum horizontal coverage zone is $\Delta\theta'_{\min} = 136°$ (from $\theta = -70°$ to $\theta = 66°$), respectively, at f'_H frequency.

In the right hand side of Figure 5.8 we show the patterns in the main vertical cut ($\varphi = 90°$, $0 \leq \theta < 360°$) at the central, lower, and upper frequencies. The minimum vertical coverage

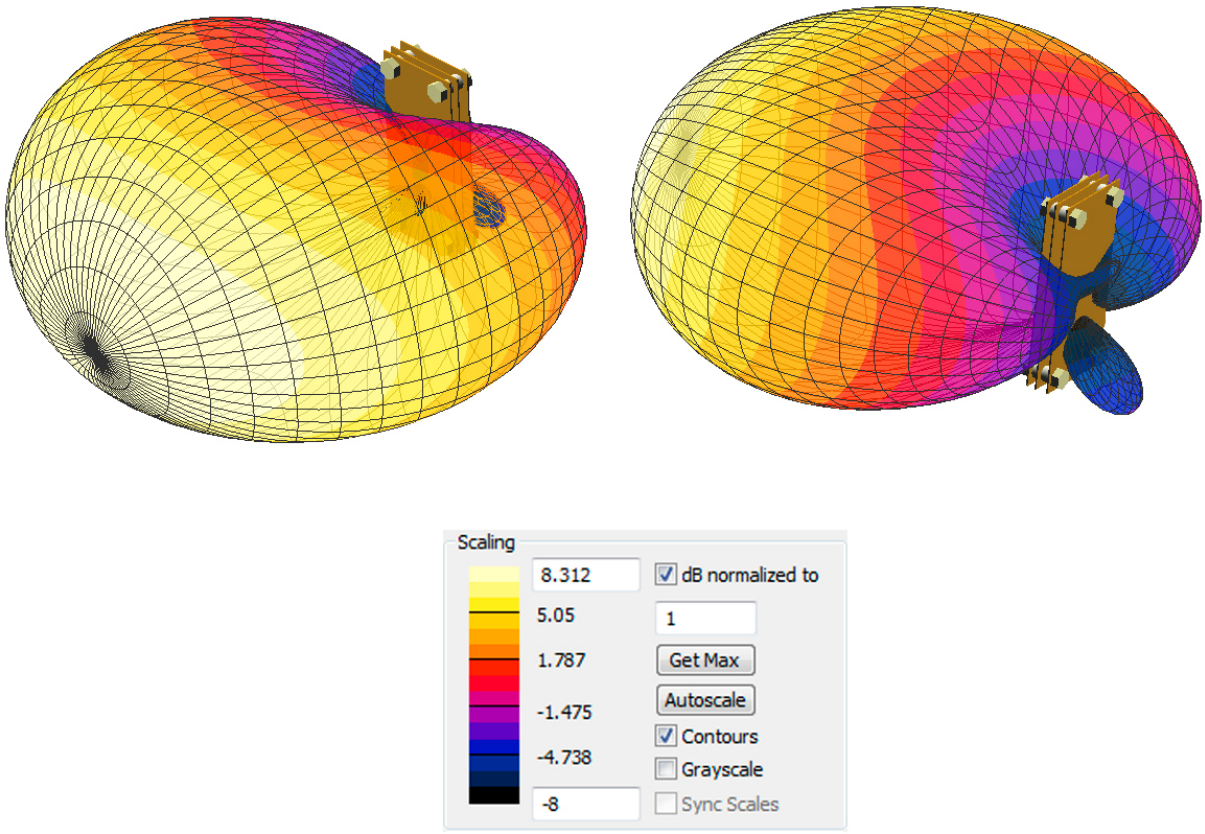
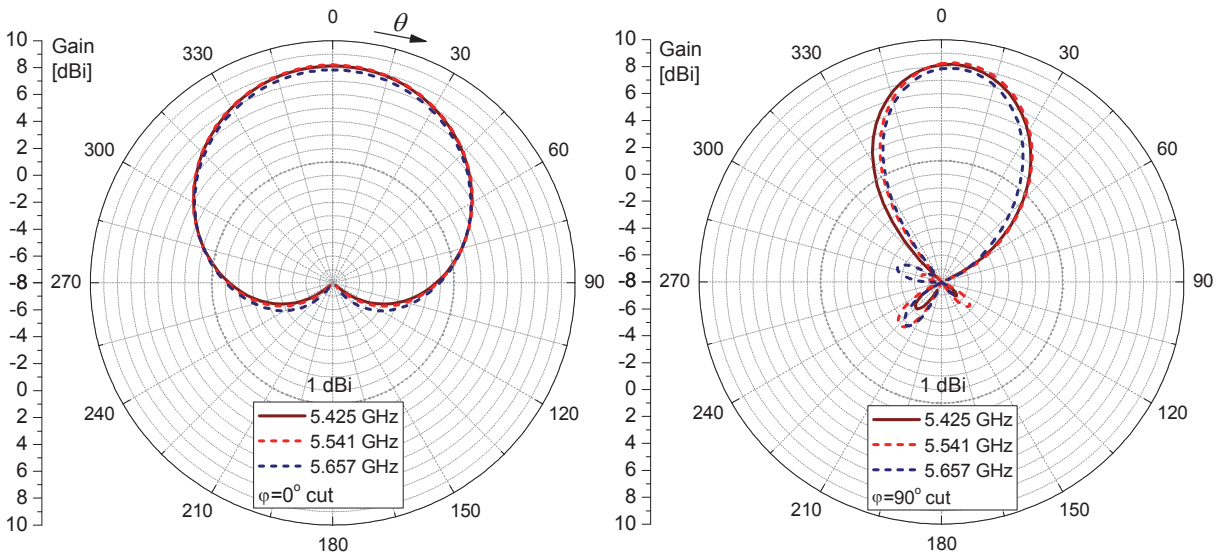
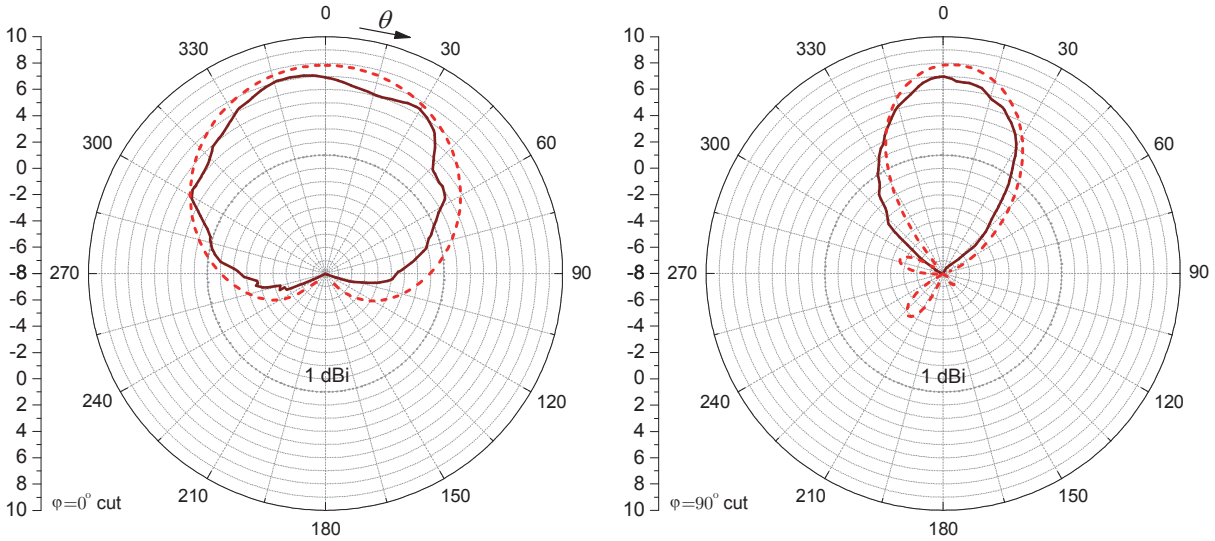


Figure 5.7: 3D normalized copolar gain pattern ($|E_y|$ component) simulated at the central frequency 5.541 GHz.



(a) (b)

Figure 5.8: Gain of the $|E_y|$ copolar field at f_C, f_L, and f_H frequencies in the $\varphi = 0°$ horizontal cut (a), and $\varphi = 90°$ vertical cut (b).



(a) (b)

Figure 5.9: Simulated copolar (red dotted line) gains at $f_H = 5.657\,\text{GHz}$ and measured copolar (red continuous line) gains at $f'_H = 5.806\,\text{GHz}$ ($\varphi = 0^o$ (a) and $\varphi = 90^o$ (b) cuts).

zone $\Delta\theta_{\text{min}} = 68°$ (from $\theta = -28°$ to $\theta = 40°$, over the $1\,\text{dBi}$ reference gain) is obtained at the f_H frequency, and is reduced as the frequency increases. The measured minimum vertical coverage zone is $\Delta\theta'_{\text{min}} = 69°$ (from $\theta = -34°$ to $\theta = 35°$), also at f'_H frequency.

Vertical polarization is guaranteed within an axial ratio $\rho_{\text{min}} > 18\,\text{dB}$ at f_C (simulated) and $\rho_{\text{min}} > 11.93\,\text{dB}$ at f'_C (measured).

In Figure 5.9 we show a comparison between the simulated and measured horizontal (a) and vertical (b) cuts of the one-patch antenna. We have only measured this first antenna at two specific frequencies, $5.495\,\text{GHz}$ and $5.806\,\text{GHz}$, which are the measured lower (f'_L) and upper (f'_H) frequencies of the antenna (where $|S_{11}|_{\text{dB}} < -10$, see Figure 5.4), and we show the comparison between the upper frequency cases, whose relative gains are closer (see Figure 5.4 about the measured gain vs. frequency and Section 5.5.2 for more details for the frequency range obtained by measuring the $|S_{11}|$ reflection coefficient of the antenna).

5.3 Three-Patch Antenna Array Design

In this section we present a numerical model for a simple, yet efficient, conformal antenna composed of three ECPs, prepared for operating at the 3.4-–3.6 GHz WiMAX range [35]. The model has a maximum gain of $6.95\,\text{dB}$, a wide horizontal coverage, and an input impedance bandwidth of 6%.

5.3.1 Numerical Model Design

As with the previous antenna, we used a simulation tool based on the FDTD method [30] to design a new antenna model, which is composed of three rectangular *RP* patches as shown in Figure 5.10, where l_{RP} and w_{RP} represent the width and length of each patch, respectively, and d_{RP} is the distance between contiguous elements. The *RP*s are electromagnetically coupled to their *SL* microstrip lines, with size $l_{SL} \times w_{SL}$. The *SL*s are connected by means of a common line of width w_{JSL} and fed by a single 1 mm diameter pin *P*. The antenna is conformed above a semi-cylindrical trigonal ground plane formed by rectangular *GR* boards. The lateral ground

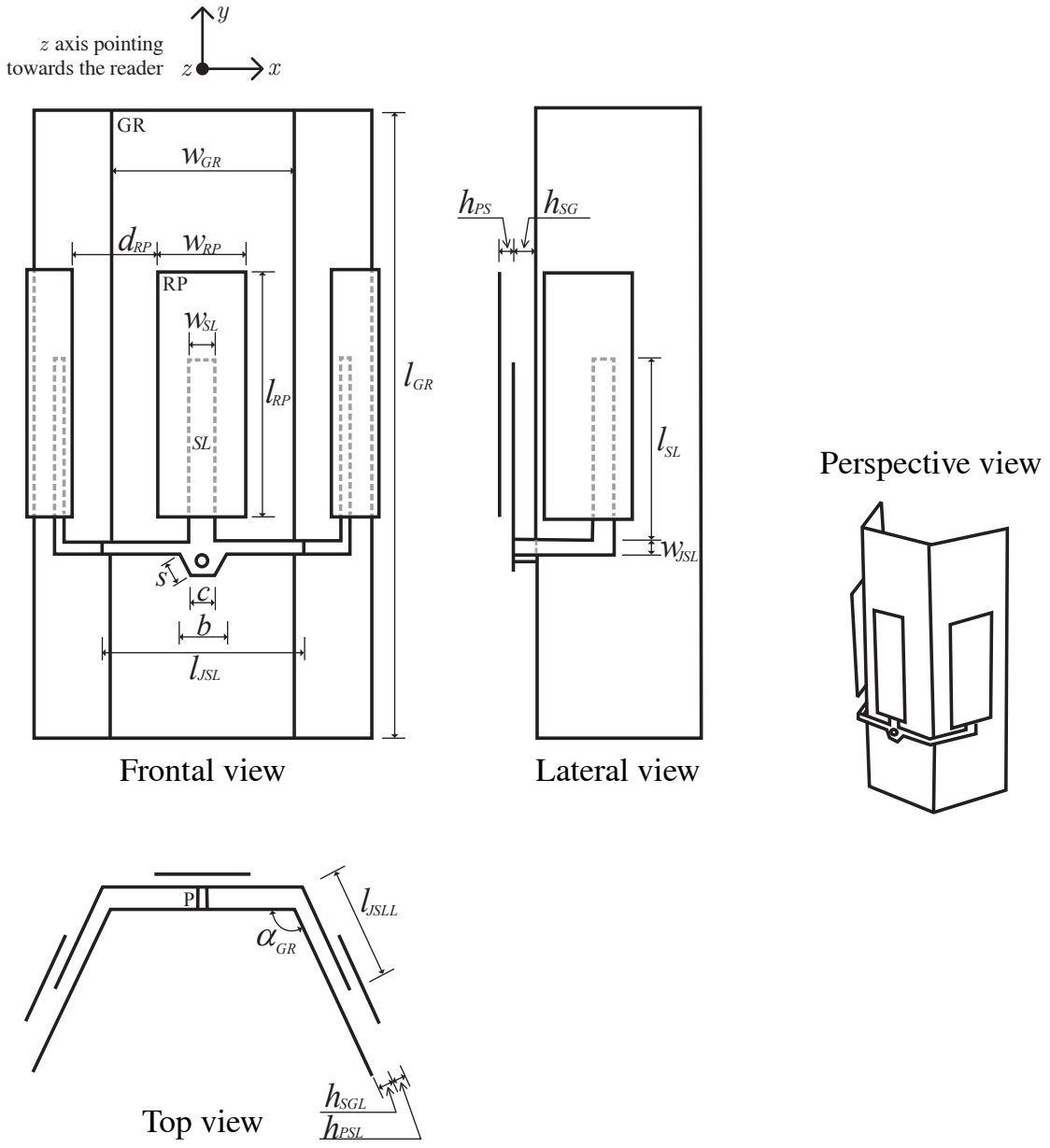


Figure 5.10: Front, side, bottom and perspective views of the geometry of the proposed conformal antenna. Relevant dimensions are also included.

plane boards are arranged in α_{GR} angles with respect to the central patch. As shown in the figure, both *RPs* and *SLs* are parallel to the corresponding *GR* boards. The model also includes a dielectric substrate with electrical characteristics similar to air ($\varepsilon_r = 1$), such as foam. This dielectric substrate is not shown in the figures for simplicity's sake.

5.3.2 Design Technique

To fulfill the requirements in Section 5.1 we follow the two steps described in Figure 5.11.

The first step is identical to the one specified for the 1-patch antenna design, described above in Section 5.2, which is mainly responsible for the bandwidth and frequency range support of the whole antenna, followed by a second step where we group three equal elements, each of them formed by one radiating patch and a common feed microstrip line. In this second step we enter a second trial error loop where we analize the combined coverage and the reflection

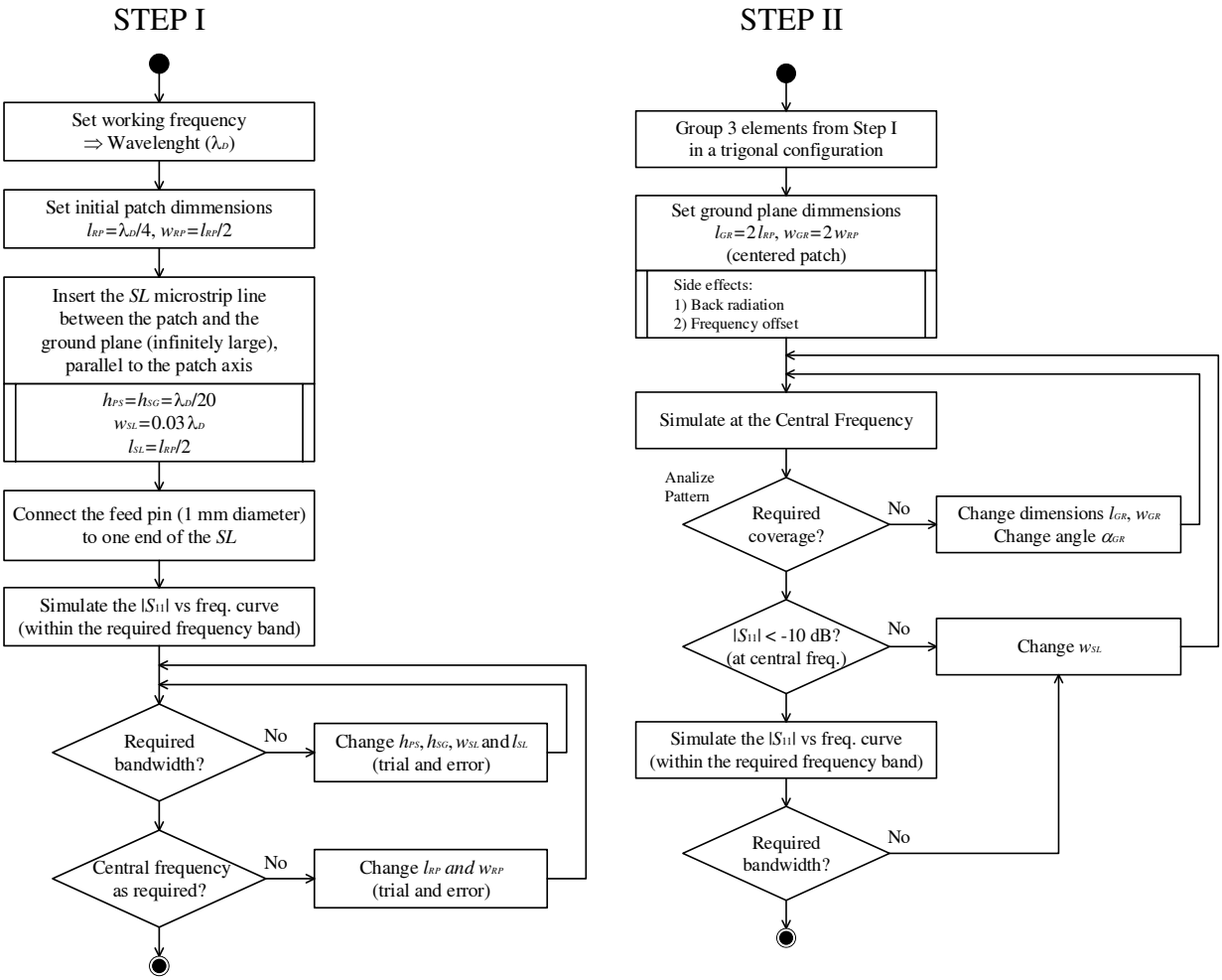


Figure 5.11: Step I of the antenna design to optimize a one-patch antenna (the central patch). Step II of the antenna design where we group three equal elements from Step I and optimize all of them together.

coefficient $|S_{11}|$. If the coverage is not enough we change the ground plane *GR* size (l_{GR} and w_{GR}) and the angle between ground planes α_{GR}, and simulate again. Next, if the reflection coefficient $|S_{11}|$ is not below -10 dB within the required minimum frequency band, we change the width of the microstrip line w_{SL}, and restart the process. Finally, we one more time test if the bandwidth is still covering the requirements, and if not we change the w_{SL} again and repeat the simulation process.

5.3.3 Results

Table 5.2 summarizes the optimized geometrical parameters according to the aforementioned requirements, as well as some design parameters.

w_{GR} [mm]	l_{GR} [mm]	α_{GR} [mm]	w_{RP} [mm]	l_{RP} [mm]	d_{RP} [mm]	w_{SL} [mm]	l_{SL} [mm]	w_{JSL} [mm]	l_{JSL} [mm]	l_{JSLL} [mm]	s [mm]
25.00	85.70	60.00	15.00	30.75	10.85	4.20	22.75	1.85	27.25	15.70	3.20
c [mm]	b [mm]	h_{PS} [mm]	h_{SG} [mm]	h_{PSL} [mm]	h_{SLG} [mm]	f_L [GHz]	f_C [GHz]	f_H [GHz]	$BW_{\%}$	$G_{\max}$ [dBi]	$\rho_{\min}$ [dB]
3.20	6.40	3.00	3.00	2.47	2.30	3.38	3.50	3.61	6.58	6.95	19.75

Table 5.2: Antenna parameters obtained by numerical simulation.

The antenna fulfills the design parameters over the whole WiMAX band (3,4–3,6 GHz). It can be seen in Figure 5.12 that the $|S_{11}|$ curve is tuned when $f_C = 3.5$ GHz.

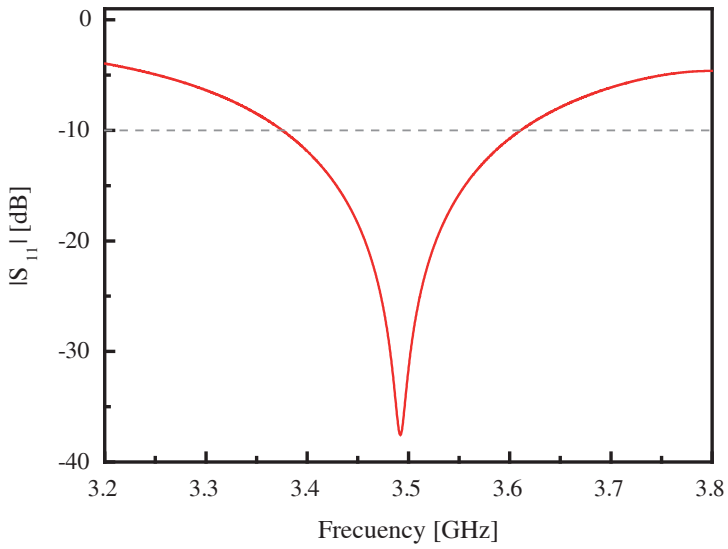


Figure 5.12: $|S_{11}|_{dB}$ versus frequency curve.

Figure 5.13 represents the 3D normalized copolar gain pattern, in dB, simulated at the central frequency of $3.50\,\text{GHz}$ (f_C for the numerical model), showing an excellent angular coverage over the $\varphi = 0°$ as well as over the $\varphi = 90°$ planes, as specified in the design requirements. The maximum gain at this frequency is $6.855\,\text{dBi}$ in absolute value (at $\theta = 0°$).

In the left hand side of Figure 5.14 we show the absolute horizontal gain pattern ($|E_y|$ component) at the lowest f_L, central f_C and highest f_H frequencies (where $f_L \leq f_C \leq f_H$, when $|S_{11}(f)| \leq -10\,\text{dB}$). In this frequency range we observe a stable pattern (with low ripple), for the whole coverage zone. The maximum gain $G_{\max} = 6.95\,\text{dBi}$ is obtained at f_L. The minimum horizontal coverage zone $\Delta\theta_{\min} = 183°$ (from $\varphi = -91°$ to $\varphi = 92°$ the absolute gain is over $1\,\text{dBi}$) is also obtained at the f_L frequency. The vertical polarization is guaranteed within an axial ratio $\rho_{\min} = 19.75\,\text{dB}$ at f_H.

In the right hand side of Figure 5.14 we show the patterns over the main vertical cut ($\varphi = 90°$, $0 \leq \theta < 360°$) at the central, lower, and upper frequencies. The vertical coverage zone, $\Delta\theta_{\min} = 68°$ (from $\theta = -34°$ to $\theta = 34°$) is obtained at the f_H frequency, and is very stable within the frequency band.

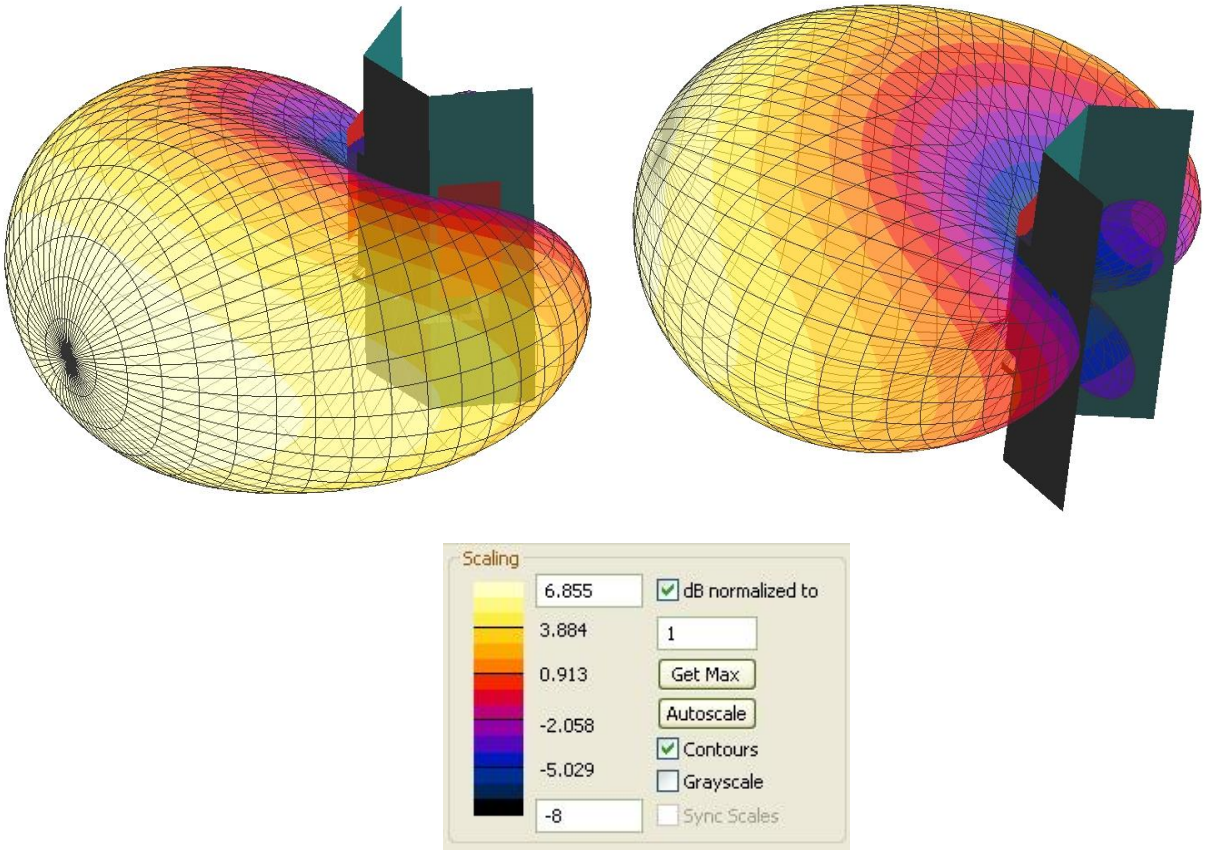


Figure 5.13: 3D normalized copolar gain pattern ($|E_y|$ component) simulated at the central frequency $3.50\,\text{GHz}$.

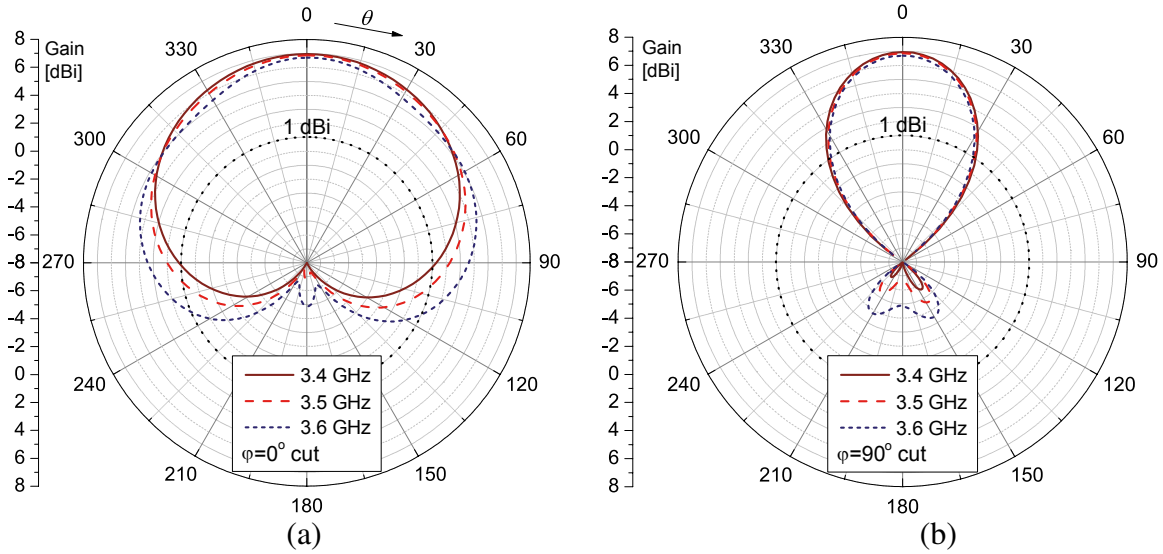


Figure 5.14: Gain of the $|E_y|$ field at f_C, f_L, and f_H frequencies in the $\varphi = 0°$ horizontal cut (a) and $\varphi = 90°$ vertical cut (b).

5.4 Five-Patch Broadband Antenna Array Design

In this section we present the design, construction and measurement of an antenna array conceived to fulfill the requirements of both the 802.11–2012 WiFi and 802.16–2012 WiMAX IEEE standards at the $5 - 6$ GHz frequency band. The proposed antenna array exhibits high gain and excellent linear polarization within a broad coverage zone.

The numerical design (see Section 5.4.1) was, as in previous sections, performed with the help of an electromagnetic simulator software tool [30], obtaining an optimized antenna array composed of ECPs [148]. The optimization technique was based upon a trial and error procedure (see Section 5.4.2). Such a numerical model has been properly verified by the corresponding fabrication and measurement of an experimental prototype (see Section 5.4.3). It will be observed that the measured electrical parameters of such a prototype are in good agreement with the simulation results: good gain (radiated along the normal to the main plane of the antenna), which varies between 5.13 and 9.37 dBi; excellent horizontal and vertical coverage zones; and a broad input impedance matching band ($|S_{11}|_{dB}$ vs. frequency curve) of about 25.6% for the experimental model (20% for the numerical one).

The antenna features (e.g. compactness, ease of connection, good coverage zone, excellent axial ratio within both the coverage zone and the frequency band) make it particularly suitable for indoor or outdoor devices where the polarization lines of the transmitting and receiving antennas are easily oriented. As will be seen, the design novelty relies on the fulfillment of all the above mentioned features at the same time, and what makes such a model even more attractive is its relatively simple geometrical configuration.

5.4.1 Numerical Model Design

The SEMCAD X simulation tool based on the FDTD [30] was used to design the numerical model of the antenna array. Such a model (see Figure 5.15), is made up of 5 rectangular patches (RP_1 to RP_5) with three different sizes that are symmetrical with respect to the central patch (RP_3): the length and width of the external ones are l_{RP1} and w_{RP1}, respectively; for the ones in between they are l_{RP2} and w_{RP2}; and finally, for the central patch they are l_{RP3} and w_{RP3}.

It is well known that the directivity (and thus the gain, provided that the efficiency is kept unchanged) of any array increases with the number of elements [94], and in this case the use of 5 elements is a compromise solution that balances compactness and high directivity (which, under appropriate values of mismatch and radiation efficiencies, implies high gain [72]). The radiating patches are fed through electromagnetic coupling to their corresponding microstrip lines of length v and width w inter-spaced a distance d_{RP} apart and centred vertically with respect to their corresponding radiating patches. Those microstrip lines are held together through a common line LM of length d and width w, fed by a single pin P with 1 mm in diameter. Centred below the RPs and LM is located a rectangular ground plane GR of length l_{GR} and width w_{GR}. All the metallic components (i.e. the RPs, LM and GR) are attached to their corresponding dielectric substrates, all of them having thickness s, length l_{DM} and width w_{DM} with relative permittivity ε_r and losses coefficient $\tan\delta$. Those substrates are spaced h_{GF} and h_{FR} apart.

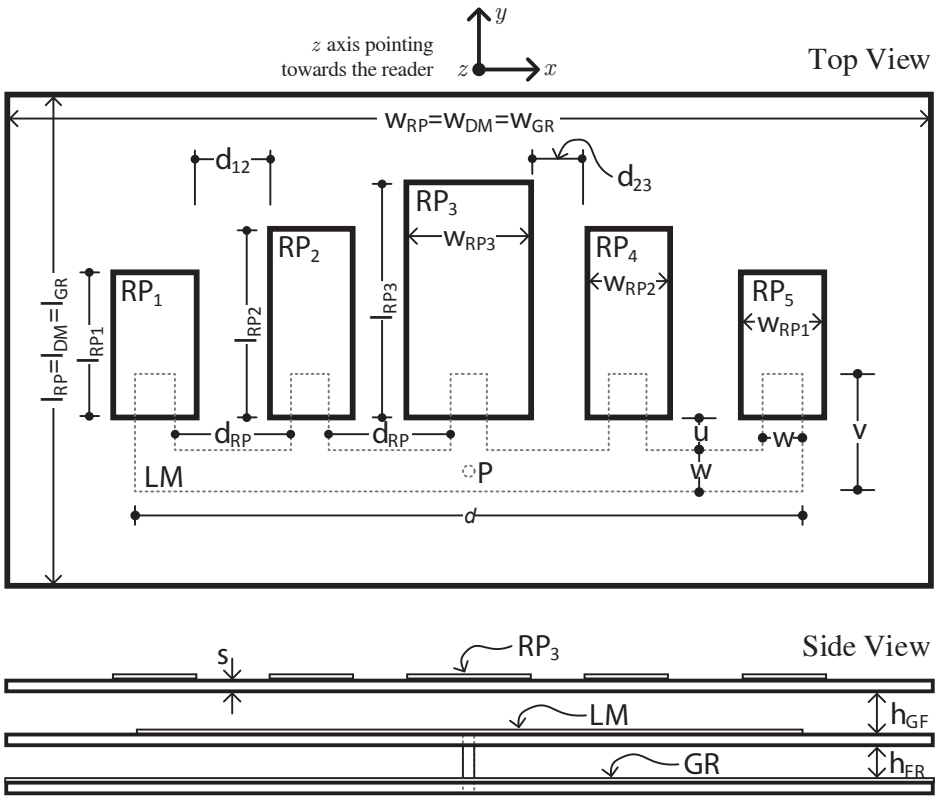


Figure 5.15: Numerical design of the antenna array.

5.4.2 Design Technique

The radiating rectangular patch of a microstrip antenna has a resonant length approximately proportional to $\lambda_D/\sqrt{\varepsilon}_r$, being λ_D the wavelength at the design (central) frequency [148]. Therefore, such a value represents an adequate option for the initial length of any radiating patch of the array. On the other hand, the widths of the patches should be short enough to ensure good linear polarization as well -but not too short, since reducing the width of a patch also reduces its fringing field [72], which in turn reduces its gain the chosen starting value thus being $\lambda_D/2\sqrt{\varepsilon_r}$. Following the dimensions of the antenna array shown in Figure 5.15, below is a list containing the initial conditions established for the model:

$$
\begin{aligned}
&\text{Set design frequency } f_D \Rightarrow \lambda_D;\\
&\text{Set } s, \sigma \text{ (copper)}, \varepsilon_r \text{ and } \tan\delta\\
&h_{FR} = h_{GF} = \frac{\lambda_D}{10\sqrt{\varepsilon_r}};\ w_{RP} = \frac{5\lambda_D}{\sqrt{\varepsilon_r}};\ l_{RP} = \frac{2\lambda_D}{\sqrt{\varepsilon_r}};\\
&w = u = \frac{0.2\lambda_D}{\sqrt{\varepsilon_r}};\ v = \frac{0.8\lambda_D}{\sqrt{\varepsilon_r}};\ d_{12} = d_{23} = \frac{0.3\lambda_D}{\sqrt{\varepsilon_r}};\\
&l_{RP3} = \frac{\lambda_D}{\sqrt{\varepsilon_r}};\ l_{RP2} = \frac{0.8\lambda_D}{\sqrt{\varepsilon_r}};\ l_{RP1} = \frac{0.6\lambda_D}{\sqrt{\varepsilon_r}};\\
&w_{RP3} = \frac{\lambda_D}{2\sqrt{\varepsilon_r}};\ w_{RP2} = \frac{0.8\lambda_D}{2\sqrt{\varepsilon_r}};\ w_{RP1} = \frac{0.6\lambda_D}{2\sqrt{\varepsilon_r}};
\end{aligned}
\tag{5.4}
$$

We found that, for the model presented here, the decreasing sizes of the radiating patches towards the lateral edges of the array constituted an appropriate configuration to produce a radiated pattern whose maximum amplitude is in the direction of the normal of the antenna[3].

After having established the initial conditions, the model behaviour is simulated and the conditions given are correspondingly evaluated. If the initial configuration does not fulfill the requirements, then the geometry is modified. The parameters to modify depend upon which of the antenna features do not match the requirements. The lengths l_{RPi} $(i = 1, 2, 3)$ of the radiating patches will in general modify the central ("resonance") frequency. They will also affect the antenna directivity (the larger the lengths, the larger the directivity). Any variation in the proportion l_{RPi}/w_{RPi} will produce a variation in the axial ratio. In this case, the larger the proportion, the better for ρ, but a too small w_{RPi} will negatively affect the electromagnetic coupling between the patches and the microstrip feeding lines, thus increasing the input mismatch. A variation in u, v and w will produce a change in the input impedance, thus producing a modification of the $|S_{11}|$ vs frequency curve. These latter parameters, together with the heights h_{FR} and h_{GF}, and the length d of LM (and thus d_{12} with d_{23}), will permit the

[3]Previous design attempts showed that, if $l_{RPi} = l$ and $w_{RPi} = w$ (for $i = 1, 2, 3$) for a broad range of l and w values, the antenna array would produce a difference power plot (i.e., a pattern with two main lobes and a null in the direction of the antenna plane normal [94]) at the central frequency.

designer to adjust the above-mentioned curve. There will be as many simulations as necessary to produce the required results, taking into account that each variable change should be in the order of $\pm 5\%$ of its corresponding value given the initial conditions, see (5.4), and $\pm 2\%$ for final refinements.

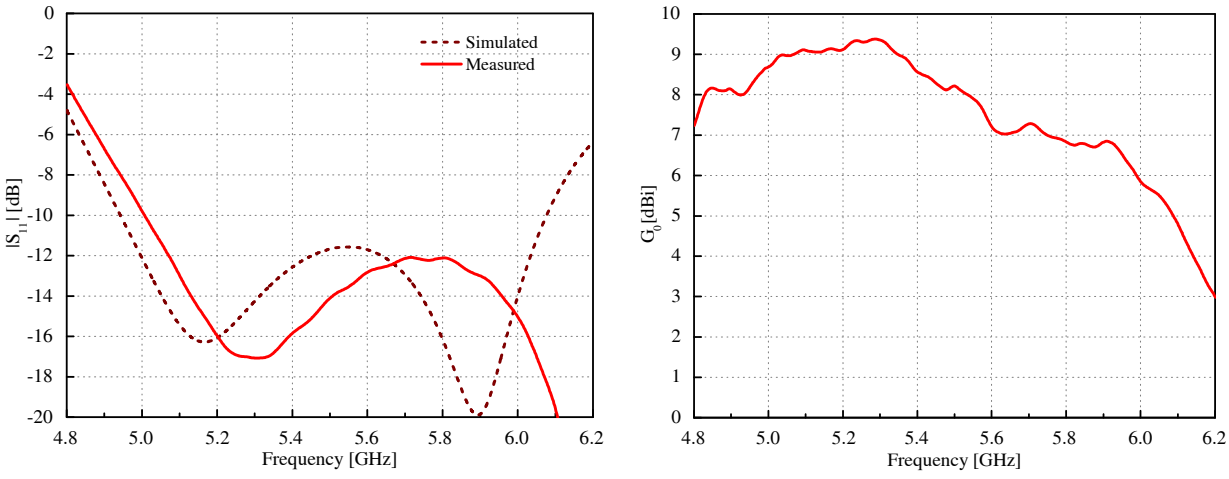
5.4.3 Results

Table 5.3 shows the values of the geometrical parameters obtained with the optimized model after having followed the procedure described in the previous section, with $\lambda_D = 54.5077$ mm (i.e., $f_D = 5.5\,\text{GHz}$). Such a table also includes the obtained antenna features, such as the antenna minimum and maximum gains, the frequency range bandwidth ($BW_{\%}$) or the $\rho_{\min}$ axial ratio.

w_{GR} [mm]	l_{GR} [mm]	h_{FG}, h_{FR} [mm]	d [mm]	d_{RP} [mm]	d_{12} [mm]	d_{23} [mm]	l_{RP1} [mm]
82.53	44.01	3.00	59.75	10.50	6.55	4.67	13.00
l_{RP2} [mm]	l_{RP3} [mm]	w_{RP1} [mm]	w_{RP2} [mm]	w_{RP3} [mm]	w [mm]	v [mm]	s [mm]
17.00	21.00	7.50	7.50	11.25	3.55	10.55	0.50
u [mm]	f_L [GHz]	f_C [GHz]	f_H [GHz]	$G_{0,min}$ [dBi]	$G_{0,max}$ [dBi]	$BW_{\%}$	$\rho_{\min}$ [dB]
3.00	4.94 (5.01)	5.51 (5.74)	6.08 (6.47)	5.08 (5.13)	9.95 (9.37)	20.68 (25.43)	> 18.00 (> 17.44)

Table 5.3: Antenna array parameters. For the meaning of the symbols, see text and Figure 5.15. The values between parentheses indicate measured parameters.

Figure 5.16 (a) shows the $|S_{11}|_{\text{dB}}$ curves obtained with the simulation FDTD tool and with the corresponding measurements. The measurements reveal a slight shift towards larger frequencies, together with a significant broadening of the bandwidth. Both effects are most probably due to the experimental value of ε_r, which surely differs from the one specified in the simulations (which was the one given in the manufacturer's manual of the dielectric used, see [28]). The frequency band shown in the figure ranges from 4.8 to $6.2\,\text{GHz}$ since the useful design band was spanned within those limits. Thus, in spite of the fact that $|S_{11}|_{dB} < -10\,\text{dB}$ comprises the range between 5.01 and $6.47\,\text{GHz}$ (see the f_L and f_H values within parentheses in Table 5.3) the frequencies above $6.2\,\text{GHz}$ have been neglected. This was done since the power pattern radiated at those frequencies becomes useless, as it has a minimum along the



(a) (b)

Figure 5.16: Reflection coefficient $|S_{11}|$ curves (a). Copolar gain $G_0 = G(0°, 0°)$, in dBi (b).

line normal to the main plane of the antenna, becoming a difference diagram [94]. When we consider the $5.01-6.47$ GHz frequency range, we obtain a bandwidth of 25.43%. In contrast, the numerical simulation gives a frequency band between 4.94 and 6.08 GHz, which corresponds to a bandwidth of 20.68%.

Figure 5.16 (b) shows the copolar gain in dBi at $(\theta, \varphi) = (0°, 0°)$ measured in an anechoic chamber over the $4.8-6.2$ GHz frequency range. As can be seen, the measured values are above 1 dBi over the whole range (see the design specifications). If we limit the band just within the useful simulation range (4.94 to 6.08 GHz, see Figure 5.16 (a)), the measured values show a maximum at 5.28 GHz whose value is 9.37 dBi, whereas the values at the extremes of the band are 8.05 dBi (at 4.94 GHz) and 5.13 dBi (at 6.08 GHz). The numerical model, however, revealed a $G_0 = 9.95$ dBi at 5.51 GHz (i.e., below the central frequency of the simulated model).

Figure 5.17 shows three different views of the experimental prototype. As in the case of the one-patch antenna prototype, the five-patch antenna array was constructed with three Rogers PCBs and an SMA connector, which was used for its connection with a $50\,\Omega$ matched transmission line. The dielectrics have been fixed through the use of M3 nylon 6–6 screws, obtaining the separation distances h_{GR} and h_{FR} by means of 0.5 mm thick M3 metal washers.

Figure 5.18 represents the 3D normalized copolar gain pattern, in dB, simulated at 5.51 GHz (f_C for the numerical model), showing an excellent angular coverage over the $\varphi = 0°$ as well as over the $\varphi = 90°$ planes, as specified in the design requirements. The maximum gain at this frequency is 9.89 dBi in absolute value (at $\theta = 0°$), which gives the 9.95 dBi mentioned above. The minimum coverage zone is $\Delta\theta_{\min} = 100°$ (for $\varphi = 0°$), and $\Delta\theta_{\min} = 79°$ (for $\varphi = 90°$), obtained at f_L and considering a minimum gain of 1 dBi as a reference. These values are slightly broadened in the measurement to $\Delta\theta_{\min} = 101°$ and $\Delta\theta_{\min} = 82°$, respectively.

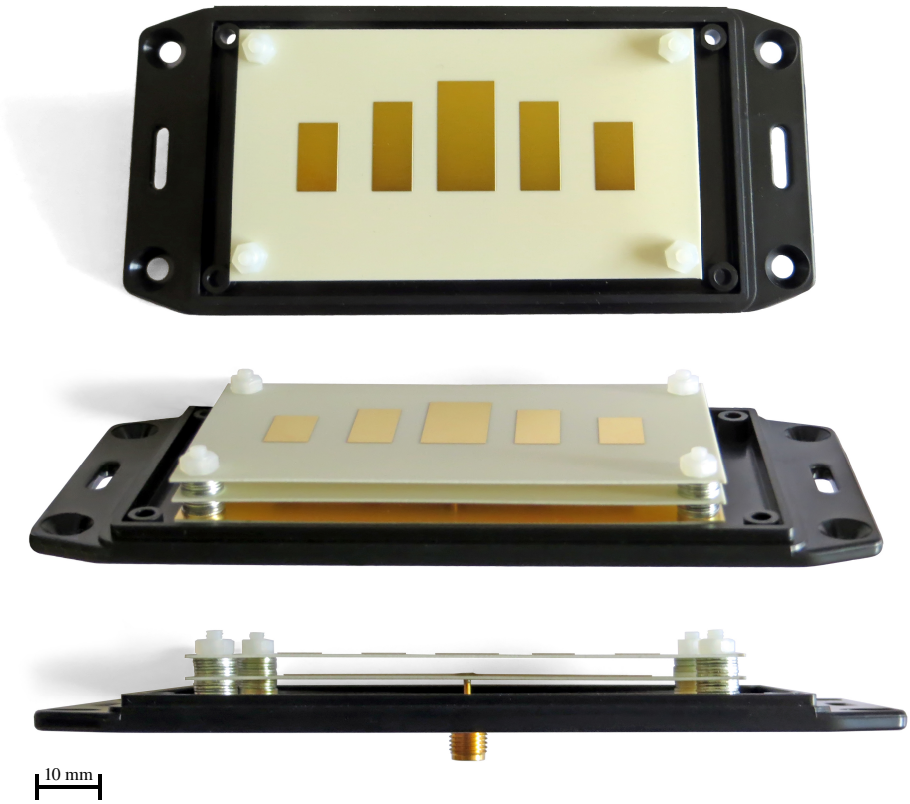


Figure 5.17: Pictures of the five-patch experimental prototype.

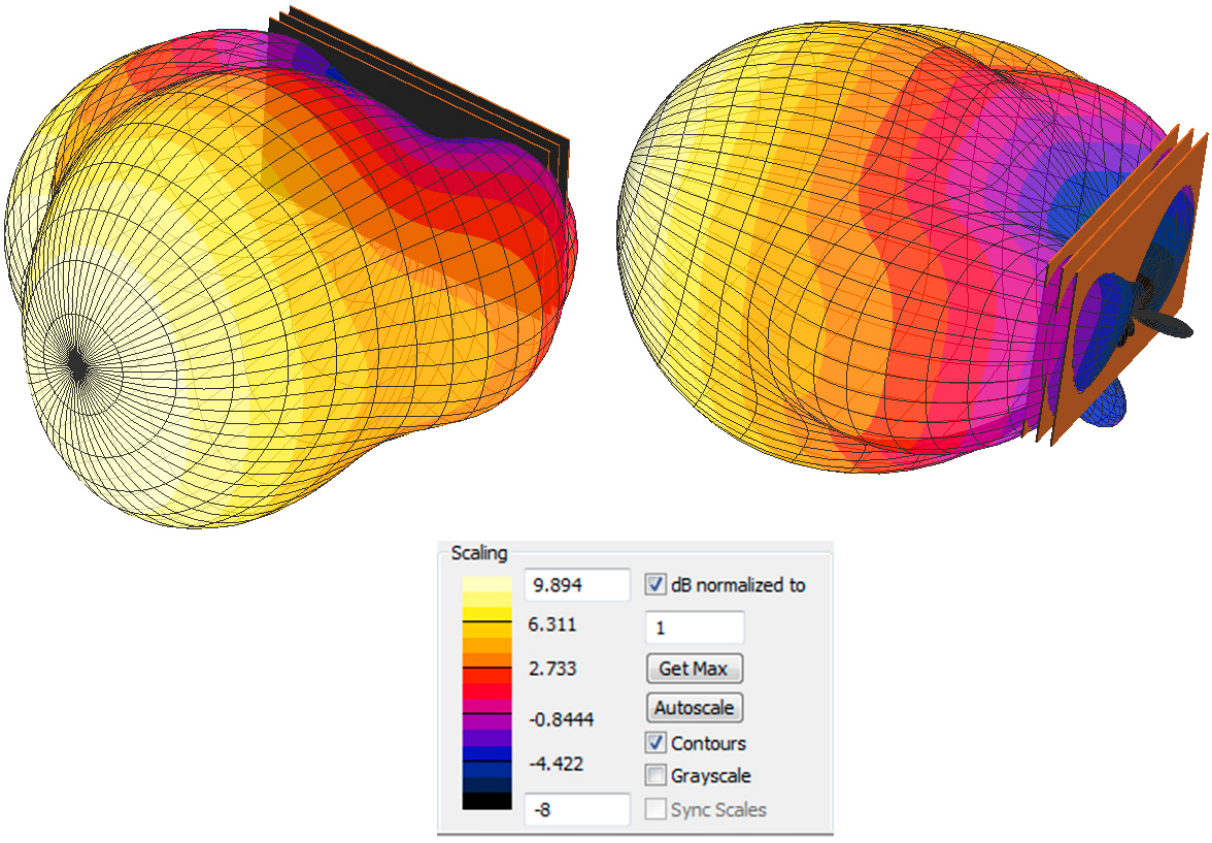


Figure 5.18: 3D normalized copolar gain pattern simulated at 5.51 GHz.

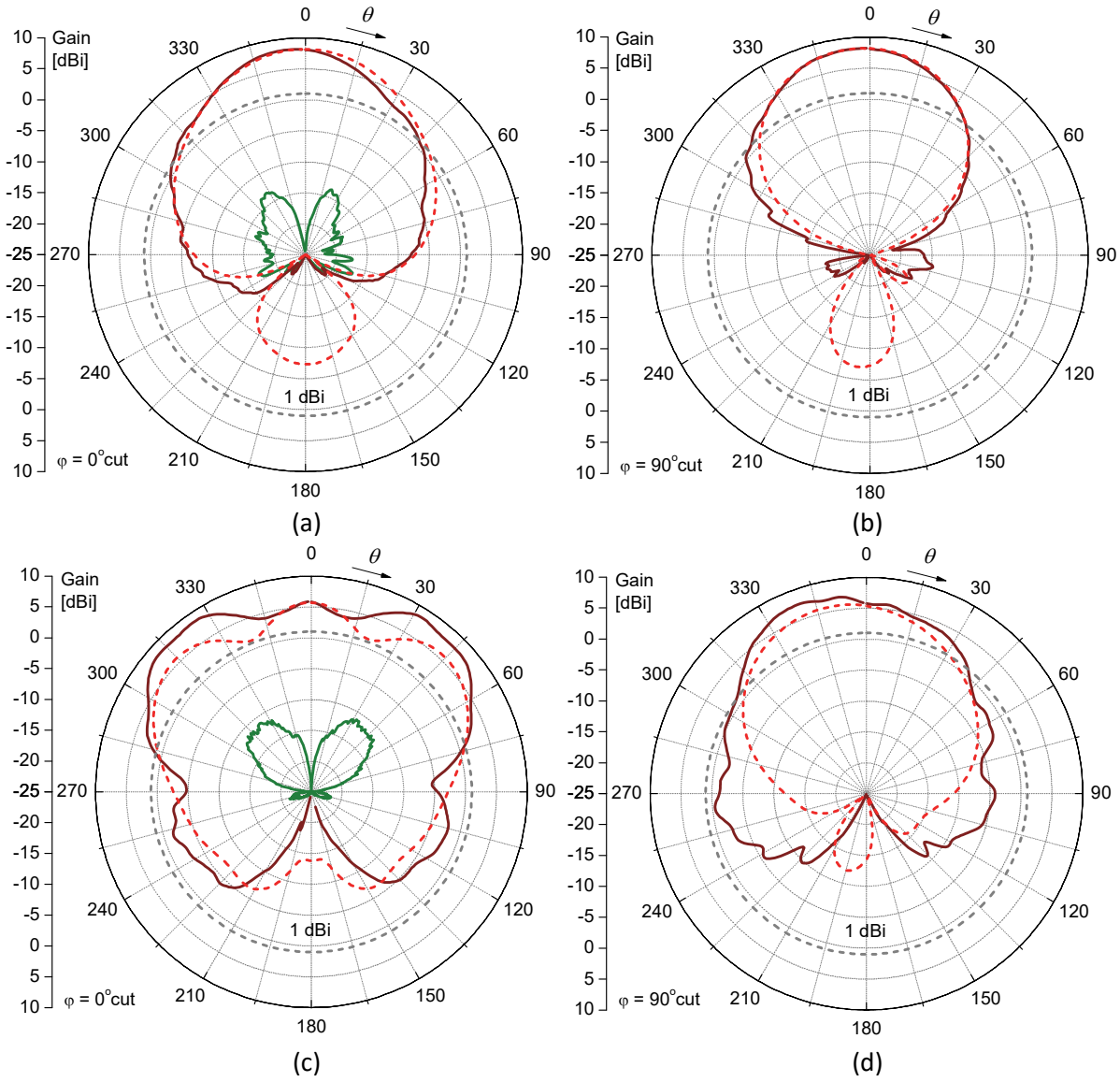


Figure 5.19: Measured copolar (red continuous line), measured contrapolar (green continuous line) and simulated copolar (red dotted line) gains at 5.00 GHz ($\varphi = 0^o$ (a) and $\varphi = 90^o$ (b) cuts), and 6.00 GHz ($\varphi = 0^o$ (c) and $\varphi = 90^o$ (d) cuts). The simulated contrapolar gains are always below -35 dBi and are not shown for simplicity. Notice also that the measured contrapolar components of subfigures (b) and (d) are below -25 dBi.

Finally, in Subfigures 5.19 (a) and (b) we show the measured (copolar and contrapolar components) and simulated (copolar) gain diagrams at $\varphi = 0°$ and $\varphi = 90°$ cuts, both having been obtained at 5.00 GHz, whist Subfigures 5.19 (c) and (d) represent the analogous curves to those given in the previous subfigures, but obtained at 6.00 GHz. In all cases the minimum value of the measured axial ratio $\rho_{\min}$ (see eq. 5.3) is above 17.44 dBi, and this can be partially verified by observing that the measured contrapolar components have relatively very low values. Since all the simulated contrapolar gains are below -35 dBi, they are not shown for simplicity.

5.5 Measurements in Anechoic Chamber

As explained in previous sections, we built the two antenna prototypes for the first (one-patch) and third (five-patch) numeric models. The build complexity of the proposed three-patch antenna model is much higher than the other two antennas. It is composed of three patches on a trigonal distribution, while the other two are composed of single PCBs, disposed in parallel. Moreover, it has a small coverage gain of only $21°$ with respect to the first antenna, that is, $183°$ vs. $162°$. For these reasons we took the decision not to build an antenna prototype in this case.

We measured the radiation parameters in an anechoic chamber of the Radioelectric Measurements Laboratory (LMR) at the Telecommunication Engineering building of the University of Vigo. The objective was to contrast the simulation results (obtained with the SEMCAD X tool [30]) with proper experimental measurements.

Such measurements were performed with adequate RF equipment and properly carried out by the professional staff of the LMR laboratory. The following equipment and installations were used:

- Agilent PNA E8361A network vector analyser.
- ETS-Lindgren 3117 antenna.
- RF structured cables and pigtails.
- Agilent 83017A RF preamplifier.
- Non-metallic antenna tripod.
- Anechoic chamber.
- Antenna positioners and positioner controller.
- Computer system for control and data processing.
- Laser distance measurer.
- Laser level measurer.

Agilent PNA E8361A Network Analyser

A high-performance and properly calibrated Agilent PNA E8361A network analyser similar to that shown in Figure 5.20 was used for the antenna measurements. Among its main features, we can indicate that it has two ports and its working frequency goes from 10 Mhz to 67 GHz [2].

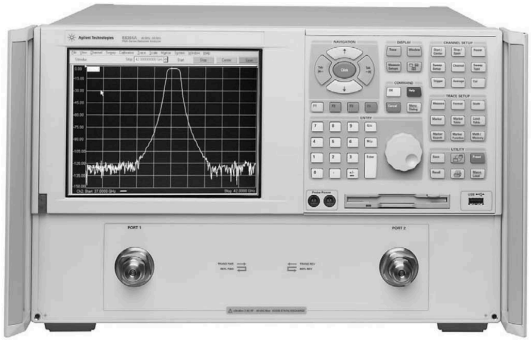
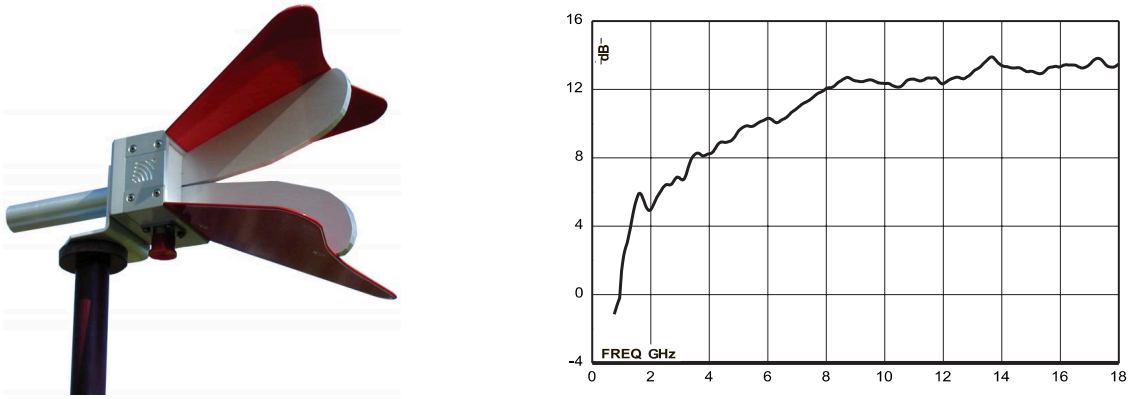


Figure 5.20: Agilent PNA E8361A network analyser.

ETS-Lindgren 3117 Antenna

For the external radiating system, two ETS-Lindgren 3117 [7] vivaldi antennas were used, which are similar to that shown in Figure 5.21 (a). They are double-ridged antennas working from 1 GHz to 18 GHz and capable of radiating linearly polarized fields [7]. One of them was used to sense the radiation diagram, while the second one was used for the gain estimation, as explained below.

It is specified by the manufacturer that this model of antenna has a polarization rejection of at least 20 dB. Due to this limitation, when using this antenna to sense the radiation diagram,



(a) (b)

Figure 5.21: ETS-Lindgren 3117 antenna (a). Antenna gain vs. frequency diagram (b).

one has to bear in mind that the system cannot deal with a polarization rejection higher than this value. Figure 5.21 (b) shows the gain versus frequency behaviour of the described antenna.

RF Structured Cables and Pigtails

High performance RF structured cables were used for the RF connection between the antenna positioner and the network analyser. Moreover, high quality RF pigtails were used for the connection between the antennas (the sensing antenna and the Antenna Under Test (AUT)) and the positioners.

Agilent 83017A RF Preamplifier

An RF preamplifier Agilent 83017A similar to that shown in Figure 5.22 (with a gain of al least 30 dB) was used to maximize the dynamic range measurement. This preamplifier was located in the base of the positioner where the AUT is installed.

Figure 5.22: Agilent 83017A RF preamplifier.

Non-Metallic Antenna Tripod

A non-metallic tripod similar to that shown in Figure 5.23 was used to hold the AUT during the $|S_{11}|$ reflection coefficient measurements.

Figure 5.23: Non-metallic antenna tripod.

Anechoic Chamber

The anechoic chamber where the measurements were taken has an overall dimension of $9 \times 7 \times 7\,\mathrm{m}$ and it is equipped with an air conditioning system capable of maintaining and controlling the ambient temperature and humidity, keeping them at $22°\mathrm{C}$ and 30% Relative Humidity (R.H.). It has metal walls covered by RF absorbing material with a maximum reflectivity of $-30\,\mathrm{dB}$ for frequencies above $500\,\mathrm{MHz}$.

Antenna Positioners

The LMR has a positioner system manufactured by Orbit/FR, on which the AUT is installed, capable of performing roll and azimut turns, as shown in Figure 5.1. The sensing antenna is attached to an additional positioner able to roll for controlling the polarization between both antennas.

Positioner Controller

To control both antenna positioners, one for the roll control of the sensing antenna and another one for the azimut and roll control of the AUT, an Orbital/FR AL-4806-3C positioner controller [51] was used, similar to that shown in Figure 5.24.

Figure 5.24: Orbital/FR AL-4806-3C positioner controller.

Computer System for Control and Data Processing

To remotely control the antenna positioner and the measuring equipment, several computers were used. They were situated in a control room beside the anechoic chamber, and they avoided the presence of unwanted objects inside the anechoic room while measuring.

Laser Distance Measurer

A precise laser distance measurer was used to determine the measuring distance of the radiation diagram and the gain between the AUT and the sensing antenna.

Laser Level Measurer

A self-leveling two-plane laser level was used in order to adjust the zero coordinates of the positioner with regard to the verticality and turn center line of the AUT.

A summary of all the above mentioned parts and their interconnections is shown in Figure 5.25.

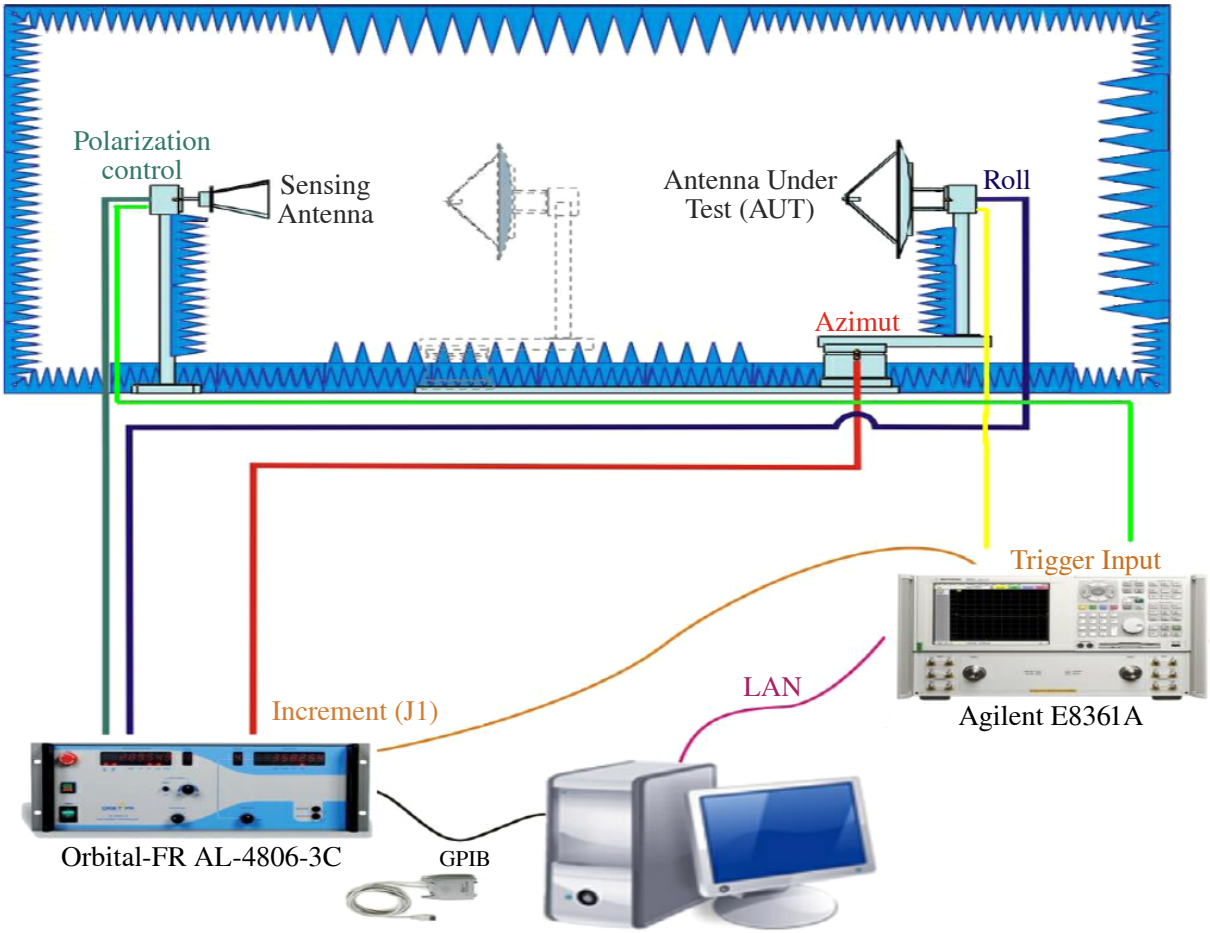


Figure 5.25: Summary of the anechoic chamber equipment and interconnections among the parts.

5.5.1 Antenna Preparation

As shown in Figure 5.26, two antenna supports were built with metal hexagonal separators and standard ABS enclosure parts, needed in order to mount the two AUTs on the antenna tripod and on the antenna positioners in the anechoic chamber. For later reference, we call the first antenna prototype (1-patch antenna) AUT1, and the second one (5-patch antenna) AUT2.

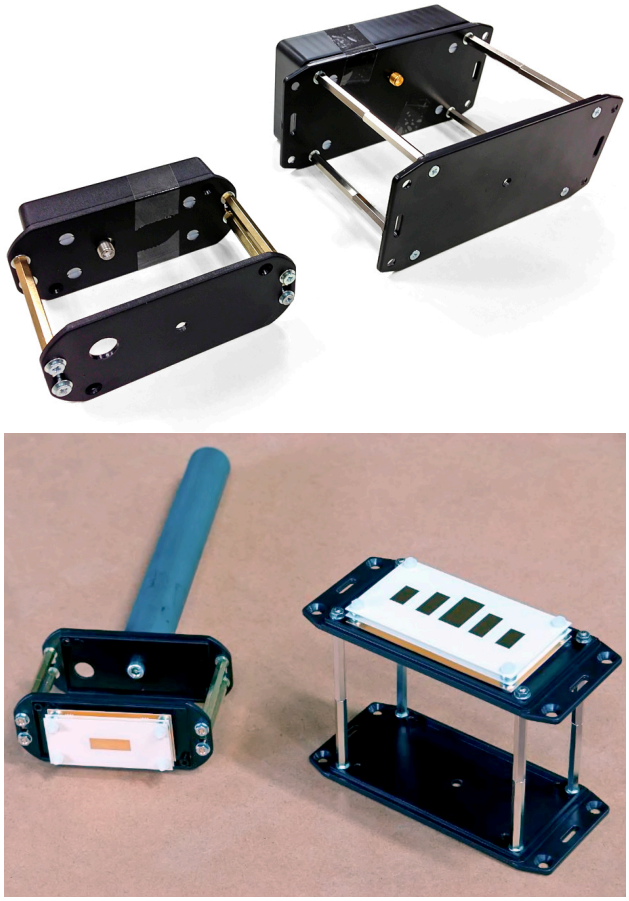


Figure 5.26: AUT1 and AUT2 antenna prototypes mounted on their support.

5.5.2 Measurement of the Reflection Coefficient

For the measurement of the reflection coefficient, we mounted the AUTs on the tripod (see Figure 5.23) inside the anechoic chamber, to minimize any environmental side effect in the measurement that could cause interference.

The network analyser, RF pigtails and adapters, the non-metallic tripod and absorbing material were used to make the measurements. Inside the anechoic chamber the network analyser was deployed on a base of absorbing material. After a 1 h period of temperature stabilization, a calibration using an RF pigtail at its end connector (directly connected to the SMA of the antenna) was performed, eliminating the need to calibrate the pigtail itself. The calibration was done using an ECal N4691B electronic calibration kit [6], to configure each antenna measurement with the parameters shown in Tables 5.4 and 5.6.

AUT1 Antenna

For the measurement of the return loss of the AUT1 we attached it to the non-metallic tripod and orientated it obliquely with respect to the walls to minimize direct reflections, at a distance of over 3 m from the walls, as shown in Figure 5.27. Once the floor has been covered with

absorbing material, we left the chamber, closed the door and proceeded with the measurement, using the configuration parameters shown in Table 5.4.

Frequency sweep method	Linear Frequency
Initial Frequency	4.5 GHz
End Frequency	6.5 GHz
Transmit Power	0 dBm
N (Number of freq. points)	1601
IF bandwidth	300 Hz

Table 5.4: Configuration parameters for the $|S_{11}|$ measurement for the AUT1.

Considering the interactions of the antenna with the environment as negligible (by following the previous antenna mounting inside the chamber), the $|S_{11}|$ parameter directly corresponds to the reflection coefficient (Γ) and return loss (R.L.) as follows:

$$\text{R.L.} = 10\ \log_{10}|\Gamma|^2 = 20\ \log_{10}|\Gamma| \cong 20\ \log_{10}|S_{11}|$$

As a result, in Figure 5.28 we show the return loss (in dB) of the AUT1, obtained from a $|S_{11}|$ measurement. And from these measurements, we can extract the frequency bandwidth for which the return losses are below -10 dB (usable bandwidth of the antenna). This is shown in Table 5.5.

For the comparison between this measurement and the $|S_{11}|$ curve obtained by simulation, see Section 5.2.

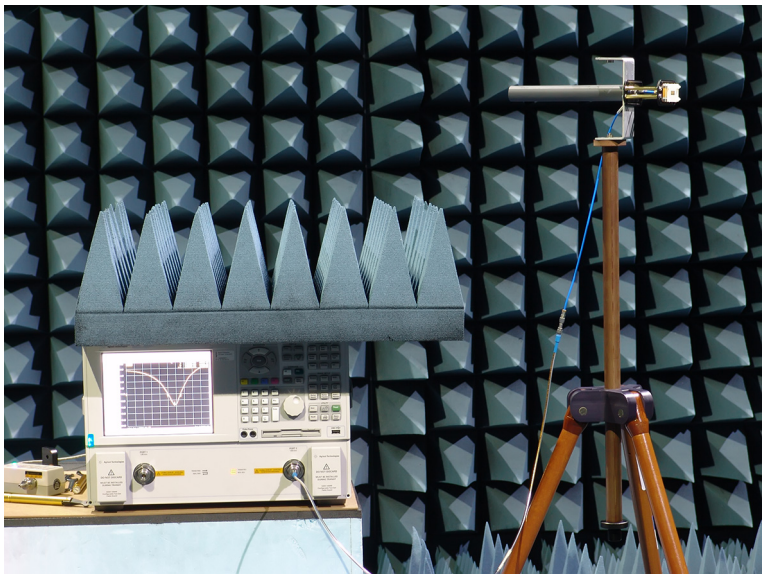


Figure 5.27: Picture of the $|S_{11}|$ measurement set-up for the AUT1.

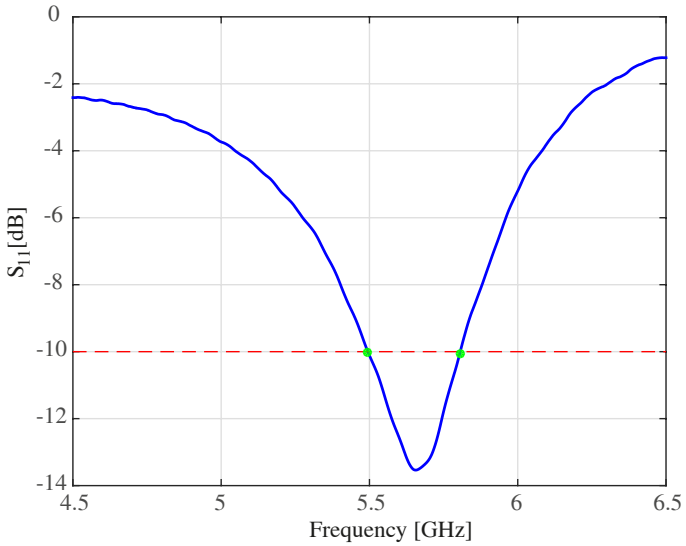


Figure 5.28: Return loss of the AUT1.

Bandwidth	310 MHz
Lower Frequency	5.495 GHz
Central Frequency	5.65 GHz
Higher Frequency	5.805 GHz

Table 5.5: Bandwidth of the AUT1.

AUT2 Antenna

For the measurement of the return loss of the AUT2 the configuration parameters shown in Table 5.6 were used.

Frequency sweep method	Linear Frequency
Initial Frequency	4 GHz
End Frequency	7 GHz
Transmit Power	0 dBm
N (Number of freq. points)	1601
IF bandwidth	300 Hz

Table 5.6: Configuration parameters for the $|S_{11}|$ measurement for the AUT2.

The measurement scenario set-up for the AUT2 is shown in Figure 5.29.

As a result, we show the return loss of the AUT2 in Figure 5.30, and the frequency bandwidth for which the return losses are below -10 dB in Table 5.7.

For the comparison between this measurement and the $|S_{11}|$ curve obtained by simulation, see Section 5.4.

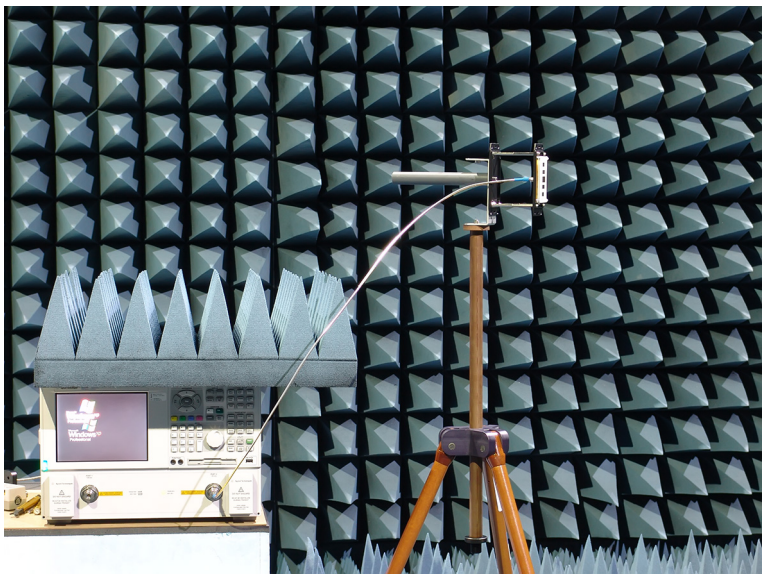


Figure 5.29: Picture of the $|S_{11}|$ measurement set-up for the AUT2.

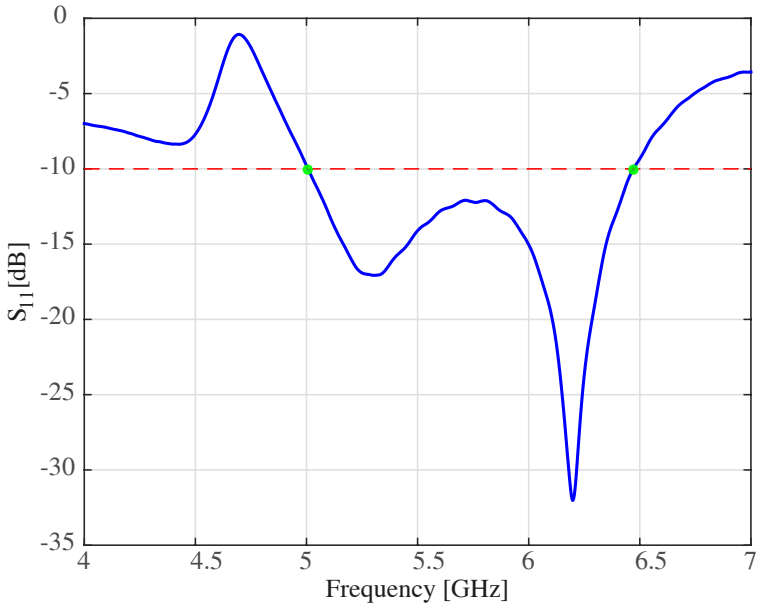


Figure 5.30: Return loss of the AUT2.

Bandwidth	1462.5 MHz
Lower Frequency	5.007 GHz
Central Frequency	5.738 GHz
Minimum S11 Frequency	6.198 GHz
Higher Frequency	6.469 GHz

Table 5.7: Bandwidth of the AUT2.

5.5.3 Measurement of the Radiation Diagram

For this measurement a positioner able to perform roll and azimut turns is needed inside the anechoic chamber. This makes it possible to explore a 3D sphere around the AUT.

The positioner system representation, sphere coordinate system and scenario set-up are the same as shown in Figures 5.1 and 5.25.

Once the AUT was placed and adjusted to the first positioner (able to roll and perform azimut turns) inside the anechoic chamber, and the ETS-Lindgren 3117 antenna was placed on a second positioner (able to roll) on the opposite wall of the chamber, we proceeded to cover the whole floor with absorbing material to minimize wave reflections, as shown in Figures 5.31 and 5.32.

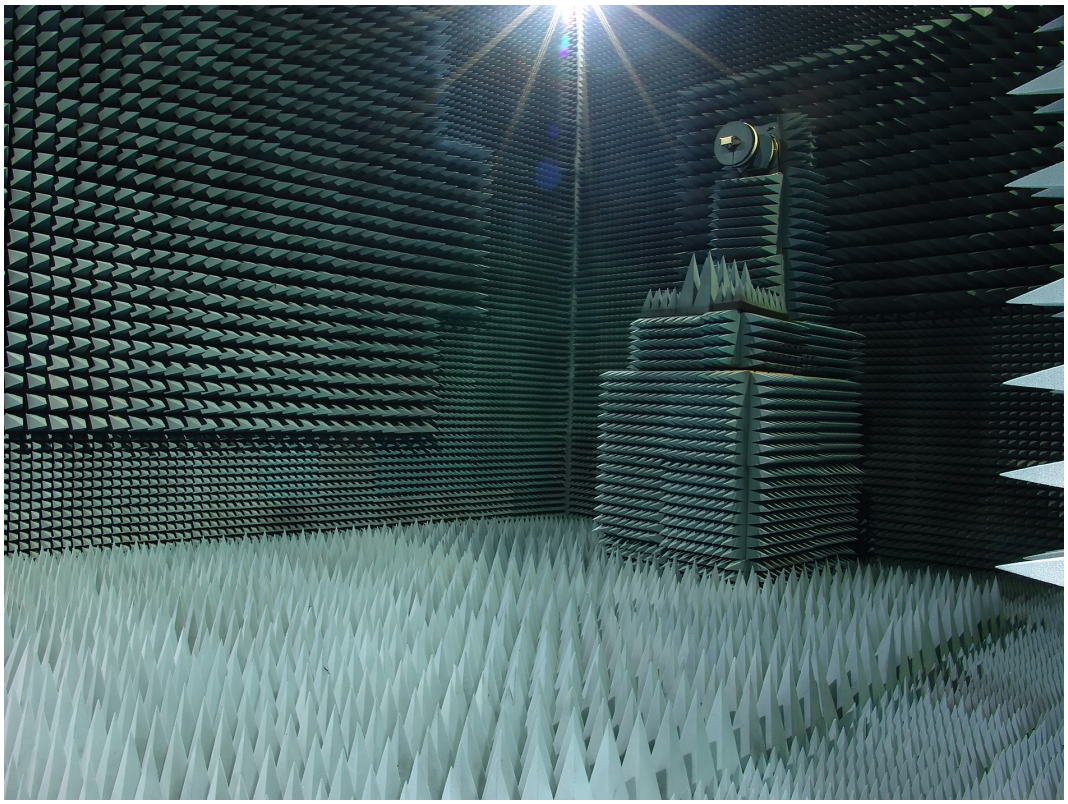


Figure 5.31: Finished AUT mount.

The measurement distance between the AUT and the ETS sensing antenna was $6.22\,\mathrm{m}$ (measured from the aperture of the ETS antenna and the top PCB of the AUT). This measurement is much higher than the near-field limit that can be estimated as [72]:

$$d_{FF} = \frac{2\,D^2}{\lambda}$$

where D is the minimum sphere diameter that contains the AUT, and λ is the wavelength at the central frequency of the AUT. Conservatively, we can specify $D = 10\,\mathrm{cm}$ for both AUT1 and

AUT2, and as the maximum frequency for both AUTs would not be higher than 7 GHz, then:

$$d_{FF} = \frac{2\,(0.1)^2}{\frac{c}{7\,10^9}} \approx 0.47\,\mathrm{m}$$

Therefore, the far field criterion is fulfilled.

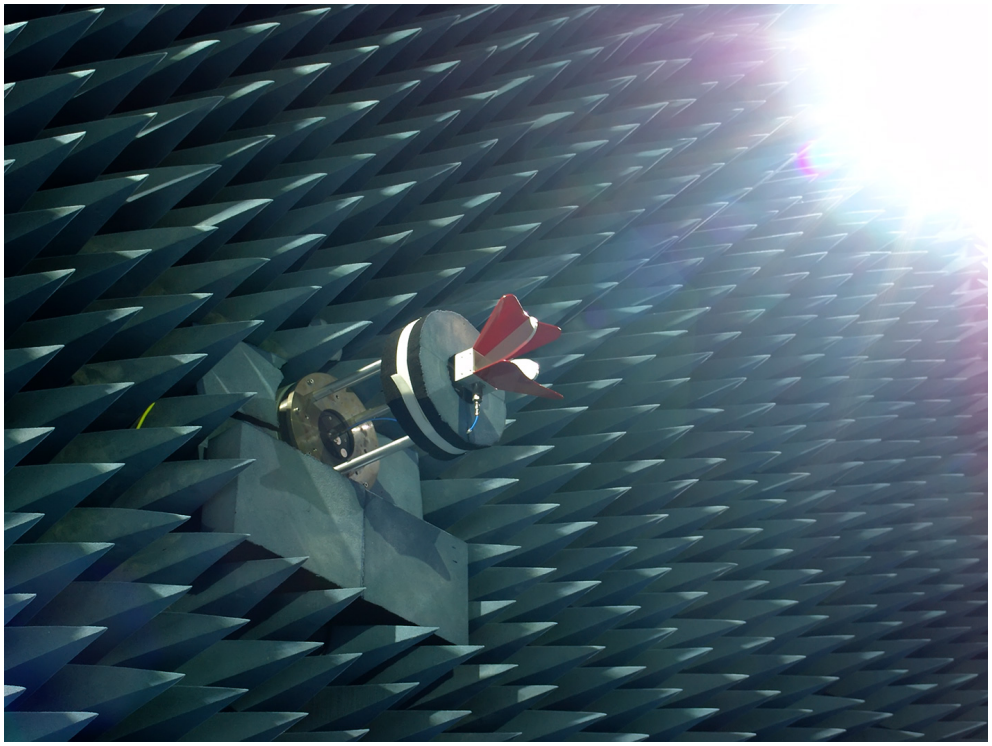


Figure 5.32: Sensing antenna view.

Using the remote control system shown in Figure 5.25 the positioners and measurement equipment were activated to perform initial tests. The sensing antenna was adjusted, with the help of the laser level measurer, to produce vertical polarization. Moreover, the AUT was adjusted vertically and on the rotation edge of the positioner in order to turn along the centre point of the top PCB of the AUT, that is, the phase centre of the antenna.

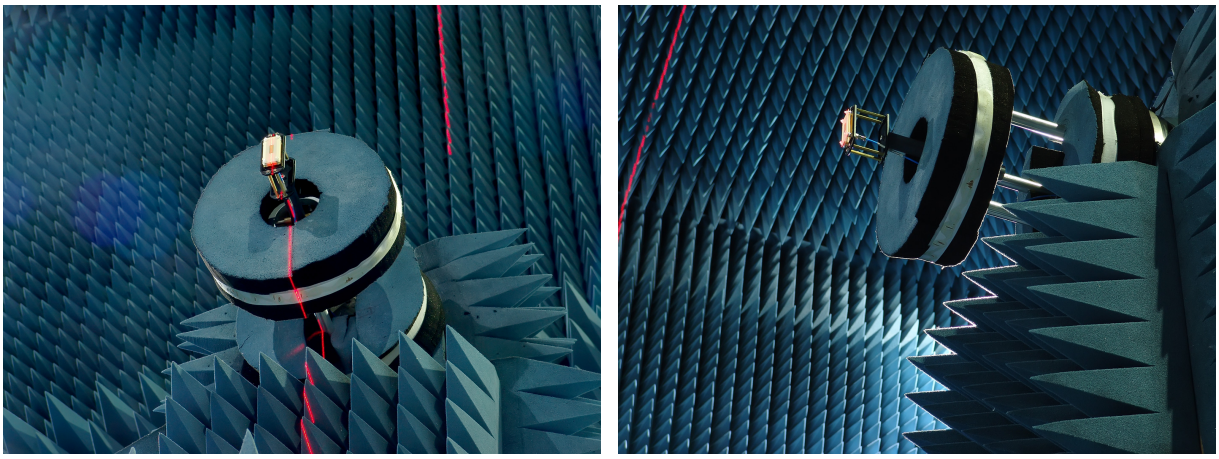
The positioner of the AUT was also initialized to be vertically polarized (copolar with the sensing antenna) with the help of the laser level. After this initial adjustment, a finer one was performed, maximizing the coupling between the sensing antenna (which was kept fixed) and the AUT (over which low scanning was performed with increments or $0.1°$). Next, the $\varphi, \theta = (0, 0)$ was fixed for this maximum coupling position.

After verifying this assembly, the whole measurement (complete sphere) was performed, with an angular discretization of $5°$ at the frequency ranges of each antenna. This process was repeated for both horizontal and vertical polarizations.

Once the measurements were finished, the data was processed to obtain the required information.

AUT1 Antenna

Prior to measuring the radiation diagram, we needed to discover which frequency to measure, this finally being around 5.65 GHz. Figure 5.33 shows how the antenna roll and tilt were adjusted and verified using the laser level, and Table 5.8 the measurement configuration.



(a) (b)

Figure 5.33: AUT1 roll adjustment (a) and tilt verification (b), using the laser level.

Measurement equipment configuration	
Frequency sweep method	Continuous Wave
Frequency	5.65 GHz
Transmit Power	−10 dB
IF bandwidth	100 Hz
Positioner sweep configuration	
First sweep	
First axis	Roll
First initial angle	−175°
First angle step	5°
First end angle	180°
Second sweep	
Second axis	Azimut
Second initial angle	0°
Second angle step	5°
Second end angle	180°
Third sweep	
Third axis	Polarization
Third initial angle	0°
Third angle step	90°
Third end angle	90°

Table 5.8: Radiation diagram measurement configuration for AUT1.

In Figure 5.34 we show a 2D and 3D (dBi) representation of the copolar component of the electrical field for the AUT1, where the maximum gain was reached at $\theta = 20.0°$ and $\varphi = 359.0°$.

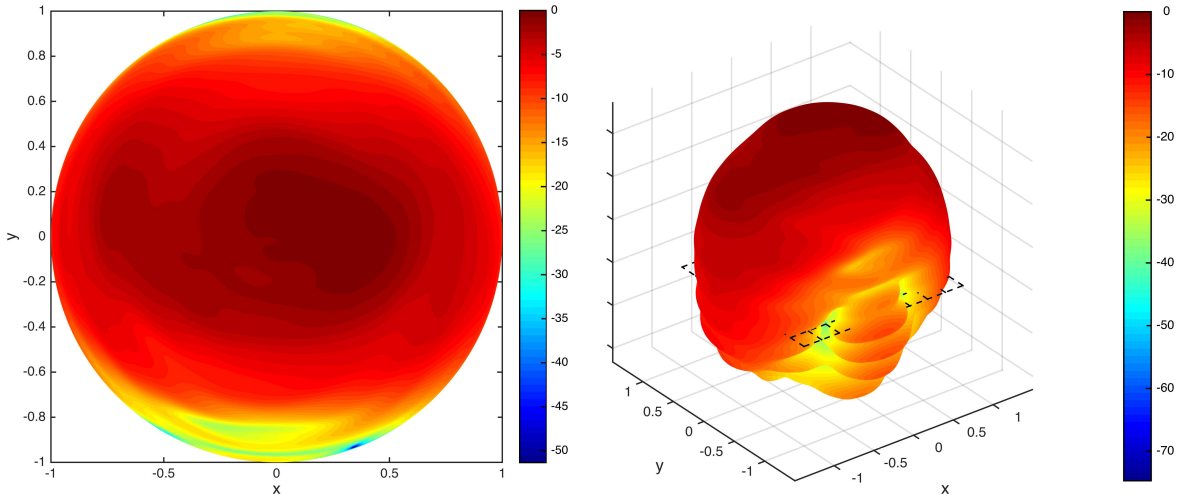
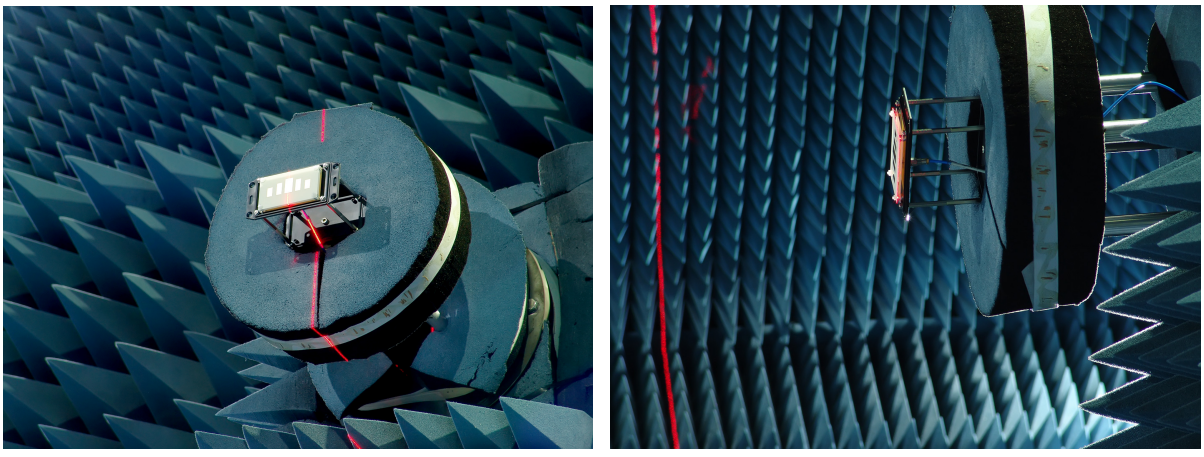


Figure 5.34: AUT1 2D and 3D representation of the copolar component of the electrical field (at 5.65 GHz). Maximum position: $\theta = 20.0°$ and $\varphi = 359.0°$.

AUT2 Antenna

As with the AUT1 antenna, before measuring the radiation diagram, we needed to discover the frequency range in which to measure. In this case we again chose the 5.65 GHz frequency, instead of either the AUT2 central frequency or the frequency where the AUT2 $|S_{11}|$ parameter is minimum (point where the matching efficiency is best). This was done in order to better compare both AUT1 and AUT2 radiation diagrams at exactly the same frequency. Note that by increasing the frequency (reducing λ), the electrical length of the antenna is increased so that the radiation diagram shape changes. Therefore, comparing two AUT measurements at



(a) (b)

Figure 5.35: AUT2 roll adjustment (a) and tilt verification (b), using the laser level.

exactly the same frequencies is always convenient and, luckily, the 5.65 GHz point lies inside the frequency limits of both antennas (where $|S_{11}|$ is below -10 dB).

In Figure 5.35 we show how the antenna roll and tilt were adjusted and verified using the laser level, and in Table 5.9 the measurement configuration.

Measurement equipment configuration	
Frequency sweep method	Continuous Wave
Frequency	5.65 GHz
Transmit Power	-10 dB
IF bandwidth	100 Hz
Positioner sweep configuration	
First sweep	
First axis	Roll
First initial angle	$-175°$
First angle step	5°
First end angle	180°
Second sweep	
Second axis	Azimut
Second initial angle	0°
Second angle step	5°
Second end angle	180°
Third sweep	
Third axis	Polarization
Third initial angle	0°
Third angle step	90°
Third end angle	90°

Table 5.9: Radiation diagram measurement configuration for AUT2.

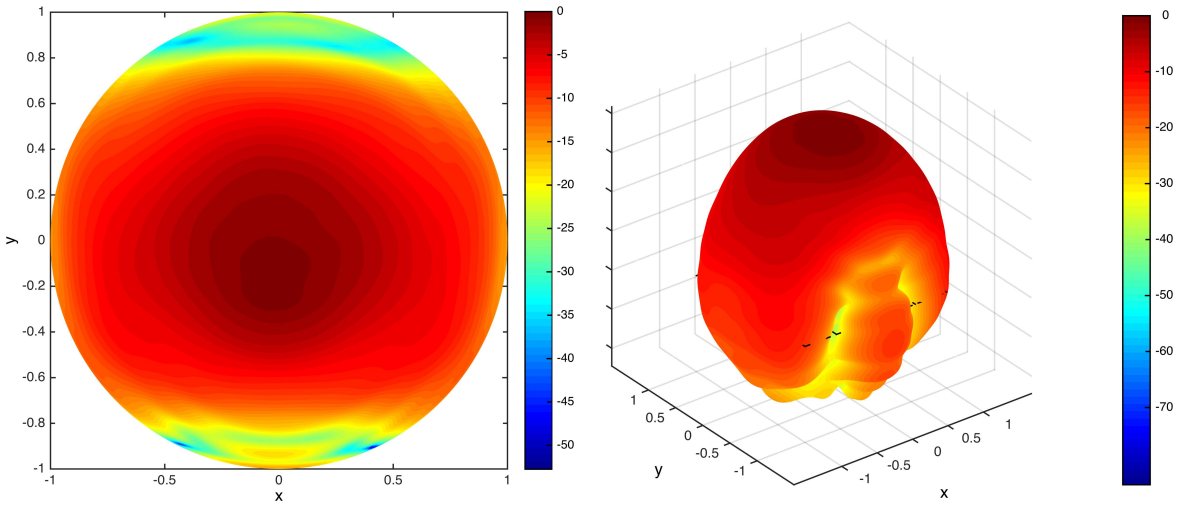


Figure 5.36: AUT2 2D and 3D representation of the copolar component of the electrical field (at 5.65 GHz). Maximum position: $\theta = 6.0°$ and $\varphi = 266.0°$.

In Figure 5.36 we show a 2D and 3D (dB) representation of the copolar component of the electrical field for the AUT2, where the maximum gain is reached at $\theta = 6.0°$ and $\varphi = 266.0°$.

5.5.4 Cut Measurements of the Radiation Diagram

We performed two main cuts for both AUT1 and AUT2 in XZ horizontal plane (i.e. $\varphi = 0°$), and YZ vertical plane (i.e. $\varphi = 90°$), at the frequency limits of each one (where $|S_{11}|$ were below $-10\,\text{dB}$). For each of them a specific roll and polarization positions were fixed for the sensing antenna and an azimut sweep was performed at the AUT positioner side.

AUT1 Antenna

For the lower ($5.495\,\text{GHz}$) and higher ($5.806\,\text{GHz}$) frequency limits of the AUT1 the configuration was set under the specifications given in Table 5.10.

Measurement equipment configuration	
Frequency sweep method	Continuous Wave
Frequency	$5.495\,\text{GHz}$ and $5.806\,\text{GHz}$
Transmit Power	$-10\,\text{dB}$
IF bandwidth	$100\,\text{Hz}$
Positioner sweep configuration	
First sweep	
First axis	Azimut
First initial angle	$-179°$
First angle step	$1°$
First end angle	$180°$
Second sweep	
Second axis	Roll
Second initial angle	$0°$
Second angle step	$90°$
Second end angle	$90°$
Third sweep	
Third axis	Polarization
Third initial angle	$0°$
Third angle step	$90°$
Third end angle	$90°$

Table 5.10: Cut configuration at the lower and higher frequencies of the AUT1.

For the comparison between these cut measurement results and the curves obtained by simulation, see Section 5.2.

AUT2 Antenna

For the lower (5.0 GHz) and higher (6.471 GHz) frequency limits of the AUT2 the configuration was as shown in Table 5.11.

Measurement equipment configuration	
Frequency sweep method	Continuous Wave
Frequency	5.0 GHz and 6.471 GHz
Transmit Power	−10 dB
IF bandwidth	100 Hz
Positioner sweep configuration	
First sweep	
First axis	Azimut
First initial angle	−179°
First angle step	1°
First end angle	180°
Second sweep	
Second axis	Roll
Second initial angle	0°
Second angle step	90°
Second end angle	90°
Third sweep	
Third axis	Polarization
Third initial angle	0°
Third angle step	90°
Third end angle	90°

Table 5.11: Cut configuration at the lower and higher frequencies of the AUT2.

For the comparison between these cut measurement results and the curves obtained by simulation, see Section 5.4.

5.5.5 Gain Estimation

The gain estimation was done in this case by comparison with a reference antenna, which is well characterized by its manufacturer. That antenna was the ETS-Lindgren 3117, identical to the one used as a sensing antenna in the previous measurements (see Figure 5.21). Once the radiation pattern measurements and the cuts at the extreme frequencies were finished for both AUTs, the $|S_{21}|$ measurement was performed at the $azimut = 0°$, $roll = 0°$ and $polarization = 0°$ (copolar) configured in the sensing antenna. After this measurement the AUT was replaced by a 3117 in the same position (as shown in Figure 5.37), maintaining an identical measurement distance. With the positioner at $azimut = 0°$ and adjusting the roll to obtain the maximum coupling with the sensing antenna (ensuring "copolarity"), then the

corresponding $|S_{21}|$ measurement was performed. By means of this procedure it was possible to obtain the gain of the AUT in the propagation (Z) axis, for $(\varphi = 0°; \theta = 0°)$.

Figure 5.37: ETS-Lindgren 3117 antenna replacing the AUT for the gain estimation.

We have to take into account the reflection coefficients of the antennas. In general, for a transmitting antenna, in natural units:

$$\widetilde{g_{TX}} = (1 - |\Gamma_{TX}|^2)\, g_{TX}$$

which, in dB notation, can be expressed as

$$\widetilde{G_{TX}} = 10\, \log_{10}(1 - |\Gamma_{TX}|^2) + G_{TX} \tag{5.5}$$

The link balance can be written mathematically in logarithmic units as follows:

$$P_{RX} = P_{TX} - L_1 + \widetilde{G_{TX}} - FSL + G_{RX} - L_2 \tag{5.6}$$

where P_{RX} is the received power, P_{TX} is the transmitted power, L_1 is the cable attenuation between the transmitter and the transmitter antenna, $\widetilde{G_{TX}}$ is the transmitter antenna gain (including the decoupling loses), FSL is the attenuation or loses due to the free space propagation, G_{RX} is the receiver antenna gain and, finally, L_2 is the cable attenuation between the receiver antenna and the receiver.

The previous mathematical statement can be related to the $|S_{21}|$ as follows:

$$S_{21} = P_{RX} - P_{TX} = \widetilde{G_{TX}} + G_{RX} - (L_1 + FSL + L_2)$$

For the general case of an AUT, denominated as AUT_i, we obtain the $|S_{21}|$ measurement as

$$S_{21,AUT_i} = \widetilde{G_{AUT_i}} + G_{RX} - (L_1 + FSL + L_2)$$

When the AUT_i antenna is replaced by the 3117 antenna, the resulting $|S_{21}|$ would be

$$S_{21,3117} = \widetilde{G_{3117}} + G_{RX} - (L_1 + FSL + L_2)$$

Subtracting both statements we get

$$S_{21,AUT_i} - S_{21,3117} = \widetilde{G_{AUT_i}} - \widetilde{G_{3117}}$$

from which we can obtain the desired AUT_i gain as

$$\widetilde{G_{AUT_i}} = S_{21,AUT_i} - S_{21,3117} + \widetilde{G_{3117}}$$

Substituting the gains with their equivalents (taking into account the antenna coefficients, as seen in Eq. 5.5), the resulting statement would be

$$\underbrace{10\ \log_{10}(1 - |\Gamma_{AUT_i}|^2) + G_{AUT_i}}_{\widetilde{G_{AUT_i}}} = S_{21,AUT_i} - S_{21,3117} + \underbrace{10\ \log_{10}(1 - |\Gamma_{3117}|^2) + G_{3117}}_{\widetilde{G_{3117}}}$$

Finally, the desired gain in dB units can by directly obtained as:

$$G_{AUT_i} = S_{21,AUT_i} - S_{21,3117} + 10\ \log_{10}(1 - |\Gamma_{3117}|^2) + G_{3117} - 10\ \log_{10}(1 - |\Gamma_{AUT_i}|^2) \quad (5.7)$$

where the terms at the right side of the statement are known or measurable.

AUT1 Antenna

As the reflection coefficient of the AUT1 was obtained between 4.5 and 6.5 GHz, it is not possible to estimate the gain outside this frequency band.

As was previously explained, once the radiation diagram was measured, one copolar measurement was performed (with $azimut = 0°$ and $roll = 0°$) with the configuration shown in Table 5.12.

Frequency sweep method	Linear Frequency
Lower Frequency	4.5 GHz
Higher Frequency	6.5 GHz
Transmit Power	0 dBm
N (number of freq. points)	1601
IF bandwidth	300 Hz

Table 5.12: Gain measurement configuration for the AUT1.

Next, the AUT was replaced by the 3117 antenna, and the measurement was performed again at the same copolar position. Finally, operating both measurements and applying the reflection coefficients of both antennas as we explained in Eq. 5.7, the sought gain value was obtained. In Figure 5.38 we show the estimated gain versus frequency curve for the AUT1.

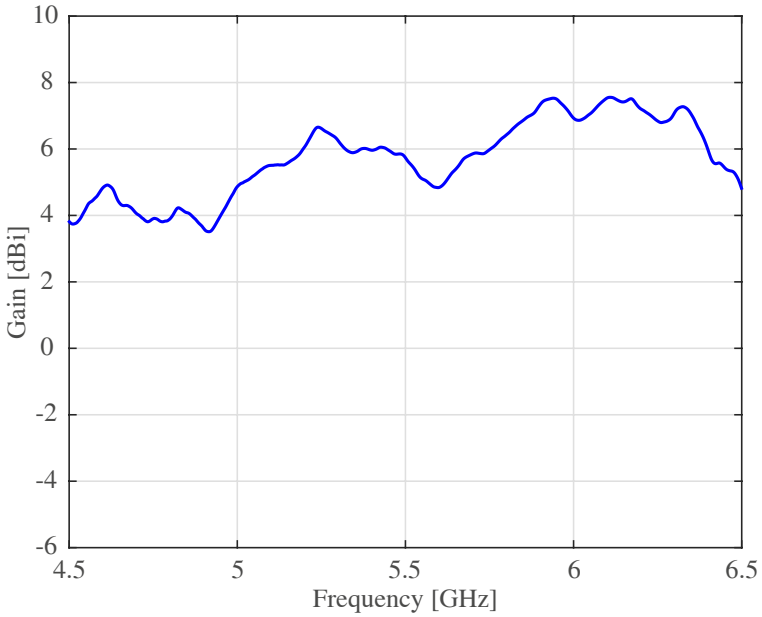


Figure 5.38: Estimated gain versus frequency curve for the AUT1.

AUT2 Antenna

For the AUT2 antenna the reflection coefficient was measured between 4 and 7 GHz, so that it was possible to estimate the gain within the whole band.

As was previously explained, once we finish measuring the radiation diagram, one copolar measurement was performed (with $azimut = 0°$ and $roll = 0°$) with the configuration shown in Table 5.13.

Frequency sweep method	Linear Frequency
Lower Frequency	4.5 GHz
Higher Frequency	6.5 GHz
Transmit Power	0 dBm
N (number of freq. points)	1601
IF bandwidth	300 Hz

Table 5.13: Gain measurement configuration for the AUT2.

Next, the AUT2 was replaced by the 3117 antenna, and the measurement was performed again. Finally, operating both measurements and applying the reflection coefficients of both antennas as explained in Eq. 5.7, the desired gain value was obtained.

In Figure 5.39 we show the estimated gain versus frequency curve for the AUT2.

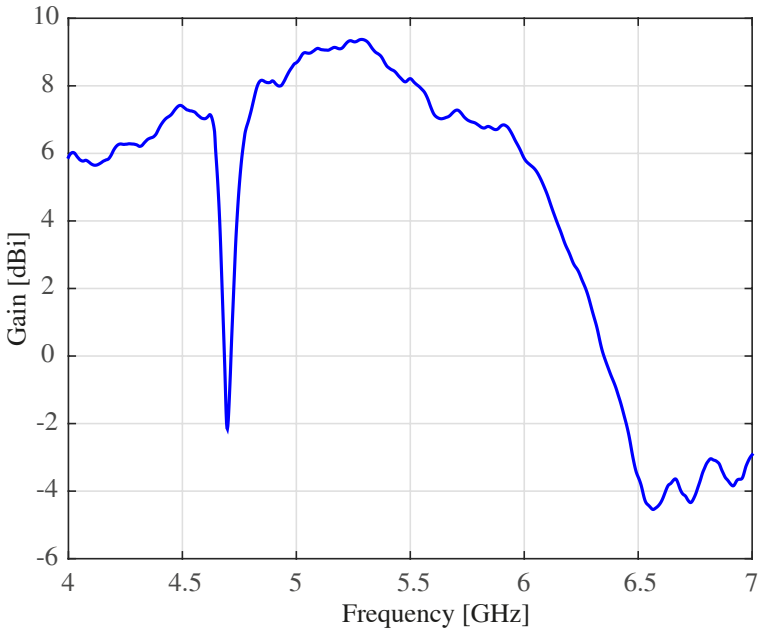


Figure 5.39: Estimated gain versus frequency curve for the AUT2.

5.6 Conclusions

In this chapter we have presented the numerical models of three antennas made up of ECPs that cover different frequency bands, compatible with both WiFi and WiMAX standards:

- One-Patch antenna: WiFi $5.47 - 5.65$ GHz band [34].
- Three-Patch antenna array: WiMAX $3.4 - 3.6$ GHz band [35].
- Five-Patch antenna array: WiFi, WiMAX $5.0 - 6.0$ GHz band.

At the time of designing each antenna, these particularly frequency bands were considered, but the proposed numerical design techniques explained for each proposed antenna model allow us to straightforwardly adapt them to any other microwave frequency range. Therefore, in principle, they could be valid for most indoor or outdoor WLANs and WSNs.

All of them also fulfill the following specified objectives: their $|S_{11}|$ vs. frequency curves were optimized for these particular frequency limits to be below -10 dB; the minimum angular coverage zones for both the horizontal ($\varphi = 0°$) and vertical ($\varphi = 90°$) cuts were maximized; and the minimum axial ratio was also maximized in order to guarantee a good vertical polarization, which is an appropriate way to increase the efficiency of the radiation transmission and, therefore, make the antenna suitable for both indoor and outdoor scenarios.

We start from a single patch ECP antenna, which is the basis of the two arrays designed afterwards (see Section 5.2 for more details).

The second antenna is an array antenna composed of three patches on a trigonal distribution, which mainly presents a 21° wider horizontal coverage than in the first case, maintaining the same radiating diagram integrity and an excellent impedance bandwidth (see Section 5.3 for more details).

And finally, we proceeded to apply the idea of making a linear array of ECPs by designing a five-patch model, in this case using a coplanar distribution of its patches on the same substrate (see Section 5.4 for more details).

For the experimental evaluation of the proposed antenna models we decided to construct some antenna prototypes and measure their performance in a proper anechoic chamber. As the build complexity of the three-patch antenna model was much higher than the other two, and because the parameter improvements of such a model with respect to the first one was not significant (mainly in terms of antenna horizontal coverage), we decided not to build a prototype of that second antenna, and only make prototypes for the one and five patch antennas. As shown in Section 5.3 the coverage gain of the three-patch antenna with respect to the first one was of 21° (183° vs. 162°, which is an improvement of only 13%), and its maximum gain is even lower than in the first case (6.95 dBi vs. 8.13 dBi).

For the construction of the two antenna prototypes, for both the one-patch and five-patch array antenna, we used 0.5 mm thick RO4000-type PCBs [28] for implementing the different substrates that compose each antenna. The main advantage is that the $\varepsilon_r = 3.38 \pm 0.05$ (relative permittivity) and $\tan\delta = 0.0027$ (tangential loss) parameters are precisely specified by the Rogers manufacturer at high frequencies and its material is specially designed for RF applications leading to more precise antenna results.

For both antennas, we disposed three PCBs in parallel, corresponding to the *RP* (radiating patch), *SL* or *LM* (transmission line) and *GR* (ground) planes (see Sections 5.2 and 5.4 for more details). With the help of a general purpose plastic enclosure (used as support), nylon screws and metal washers (with a precisely specified thickness of 0.5 ± 0.05 mm), we were able to maintain a stable orientation and precise separation of the three PCBs that compose the antennas. And by restricting the antenna models (during simulation and design) to 0.5 mm thick substrates, plus air-gaps with a similar thickness multiple of 0.5 mm, we could guarantee a precise antenna prototype construction. This helps to minimize errors due to human manipulation during their construction, by hand. Note that both the nylon screws and metal washers were proven by simulation to not interfere at all with the antenna radiation behaviour, neither to modify the $|S_{11}|$ vs frequency curve. As we indicated above, such a lack of influence could be justified by the fact that such screws were placed below the *RP* plane, placed quite far from the radiating patches, at the corners of the PCBs.

To guarantee a good antenna connection with a $50\,\Omega$ matched transmission line, we used a high-quality SMA connector (as feed point), which was properly soldered to the opposite side of the *GR* PCB, and which has good grounding connections with the opposite copper layer.

All these factors (high-quality RF materials, precision thickness parts and precise mounting technique), produced high quality and very robust antenna prototypes which would be compatible with a mass production manufacturing process. For this process we would optimize the costs by using fixed-height non-metal separators instead of the washers and the nylon screws, to quickly and accurately fix the PCBs together, with the right orientation and separation. All these accessory parts would be inexpensive.

In Table 5.14 we summarize the main antenna numeric model characteristics obtained by simulation and in Table 5.15 those obtained by measurements (for the one-patch and five-patch constructed antenna prototypes).

Antenna model	f_L [GHz]	f_C [GHz]	f_H [GHz]	G_{max} [dBi]	$BW_{\%}$	$\rho_{\min}$ [dB]	$\Delta\theta_{\min}$ ($\varphi = 0^o$)	$\Delta\theta_{\min}$ ($\varphi = 90^o$)
One-Patch	5.43	5.54	5.66	8.312 (at f_C)	4.19	> 18.00	162	68
Three-Patch array	3.38	3.50	3.61	6.95 (at f_L)	6.58	> 19.75	183	68
Five-Patch array	4.94	5.51	6.08	9.95 (at f_C)	20.68	> 18.00	100	79

Table 5.14: Summary of the three antenna model parameters obtained by simulation.

Antenna model	f_L [GHz]	f_C [GHz]	f_H [GHz]	G_{max} [dBi]	$BW_{\%}$	$\rho_{\min}$ [dB]	$\Delta\theta_{\min}$ ($\varphi = 0^o$)	$\Delta\theta_{\min}$ ($\varphi = 90^o$)
One-Patch	5.50	5.65	5.81	6.97 (at f_H)	5.49	> 11.93	136	69
Five-Patch array	5.01	5.74	6.47	9.37 (at 5.28 GHz)	25.43	> 17.44	101	82

Table 5.15: Summary of the one-patch and five-patch antenna model obtained by measurements.

It is important to note that the five-patch array antenna has a bandwidth $> 20\%$, which makes it perfectly valid for Ultra Wide Band (UWB) applications. In the particular case of the manufactured prototype, we measured a bandwidth of 25.43%, that is, a 1.46 GHz wide band over the frequency range of $5.01 - 6.47$ GHz. It also presents a minimum gain of 5.13 dBi and maximum gain of 9.37 dBi which are excellent for such a small antenna, measuring only 82.5 mm.

Combining this excellent linear-array ECP antenna performance with the small observed differences between the simulation and experimental results, we can conclude that both the proposed design technique and construction process (with the specially selected materials) used to design and respectively build the prototypes led to very good results, outperforming single-patch ECP antennas.

Chapter 6

Conclusions and Future Work

The aim of this work concerns the design and implementation of software architectures suitable for solving complex multi-technology Real-Time Location Systems (RTLSs), but we also contributed with different WSN physical layer implementations and experiments. Moreover, due to collaborations with other research groups at different universities we have contributed with a customized hardware and software solution for localization based on RFID technology, as well as with the design of new antenna models based on linear-arrays of Electromagnetically Coupled Patchs (ECPs), valid for improving Wireless Sensor Network (WSN) communication performance.

The main contributions of this work are shown in Chapters 2, 3, 4 and 5, as follows:

1. The first contribution is shown in Chapter 2 where we introduced a distributed multi-layer architecture for implementing RTLS systems. It only supports a single wireless technology at a time and is based on a fixed WSN of anchors locating mobile devices and by only using RSS measurements. The raised architecture is divided into four layers: A *Physical* layer to access the WSN at low level; a *Location* layer to implement a location algorithm (which can remotely access the *Physical* layer), and finally, another two layers (Proxy and Wrapper) to deal with the remote calls and communications between the previous two layers.

   The main benefits achieved are the re-usability and independence of the implemented layers, which can use different programming languages to be implemented, and the possibility of remote access to different WSN hardware from multiple *Location* clients.

   We also showed two possible ways of implementing a physical layer based on Bluetooth and ZigBee wireless technologies, as well as different implementations and experiments for both technologies:

   - For the Bluetooth technology we applied the proposed architecture for the experimental evaluation of a Bayesian filtering method, for jointly estimating the

target position and the path-loss exponent of a propagation model. By running different experiments we showed the effect of choosing a different path-loss exponent value for an arbitrary anchor, and how the Particle Filter (PF) could automatically adapt to a propagation change between the LOS and NLOS situations while, simultaneously, the target position is being estimated.

- For the ZigBee technology we applied the proposed architecture for the experimental evaluation of Bayesian filtering methods, in this case for target tracking in mixed indoor/outdoor environments using the ZigBee and GPS technologies.

  The experimental results present three different trajectory tracking examples for a moving target, showing the difference between only using GPS for the estimations, via a Kalman filter, versus the case where the ZigBee RSS observations are combined with the GPS positions, via a Sequential Importance Sampling (SIS) algorithm. Finally, some simulated results show a hybrid indoor/outdoor tracking experiment using the two different technologies when they are available at different time instants, so that they are used sometimes individually, and sometimes combined.

- For the ZigBee technology we also proposed a method for increasing the number of RSS available measurements, by obtaining cross RSS information among all the nodes in a ZigBee network.

  With extra RSS measurements obtained among all the anchor links, we could improve location algorithms performance by possibly using different dynamic recalibration techniques or propagation model parameter adaptation.

  We show how this method can be implemented in only three steps using commonly used prototyping hardware such as an Arduino and XBee. All the logic and output frame format presented is fully compatible with the *Physical* layer formatting proposed in this first architecture.

2. The second contribution is shown in Chapter 3 where we introduced an RTLS architecture supporting multiple wireless technologies simultaneously, multiple coordinate and mapping systems, low-level and high-level data fusion, protection and security for both data and user access, among other features. This architecture was designed to provide an easy mean of including new hardware, new algorithms and any kind of Location Based Service (LBS) by considering client applications of the location estimations of targets of interest.

   The architecture also provides a complete API with several methods for interfacing with external clients. Hardware providers can set up network nodes and technologies available

transparently, for example, to the algorithm developers. Algorithms can use raw data from one or more available technologies and nodes to provide location estimations. And these location estimations are available in real-time (or off-line, with access to historical data) for location application clients.

The main advantage of the proposed architecture, as opposed to monolithic solutions, is that it allows a generalization and retargeting for solving new LBS and applications, reducing costs of further development. Its dynamic technology support with customized sensor data can solve, in principle, any kind of multi-technology RTLS system. This solution is the opposite to other kind of purpose-built systems which can be expensive and hard to implement in practice.

To demonstrate the application of the architecture, we introduced two case studies based on real deployments. First, a multi-technology target device based on ZigBee, UWB and an accelerometer sensor. And second, a WiFi fingerprinting scenario implemented using a conventional Android smartphone. Finally, the performance of the platform for tracking a multi-tecnology (ZigBee and UWB) target has been demonstrated with validation results. They were executed with synthetic data measurements generated from a high-level simulation environment (with Matlab), making use of the real API interfaces of the platform, using actual network calls.

3. The third contribution of the Thesis is presented in Chapter 4, which shows a hardware and software solution to improve diversity of RFID readers, which can be used to implement cheaper RFID based RTLSs, covering wider areas in indoor scenarios. The solution introduced is based on an infrastructure of RFID UHF readers (following the ISO 18000-6C EPC Gen2 standard), and presents a good trade-off between hardware and software complexity while minimizing costs. The proposed RF switches multiply the number of antennas available in RFID readers by a factor of four, and they are self-powered and controlled using their GPIO port. The cost comparison results demonstrate that our solution is cost effective.

4. Finally, the fourth contribution is shown in Chapter 5, which focuses on new antenna model design, based on linear-array topologies of ECPs. We presented three numerical models that cover different frequency bands, compatible with both WiFi and WiMAX standards. We also proposed a numerical design technique for each antenna that allows them to be straightforwardly adapted to all kind of frequency ranges. Therefore, in principle, they could be valid for most indoor or outdoor Wireless Local Area Networks (WLANs) and WSNs.

   We start from a single patch ECP antenna, which is the basis of the other two array antennas. The second antenna is an array antenna composed of three patches on a trigonal

distribution, which mainly presents a wider horizontal coverage than in the first case. And finally, we designed a five-patch model which uses a coplanar distribution of its patches, on the same substrate.

For the experimental evaluation of the proposed antenna models we decided to construct some antenna prototypes and measure their performance in a proper anechoic chamber.

During the antenna prototype construction, for both the one-patch and five-patch array antenna, we used Rogers RO4000-type PCBs which were disposed in parallel and attached to a plastic enclosure (used as support), with the help of nylon screws and metal washers (with a precise thickness). We were thus able to maintain a stable orientation and precise separation of the three PCBs that compose the antennas, minimizing the errors due to human manipulation during their construction. All these factors (high-quality RF materials, precision thickness used parts and precise mounting technique), produced high quality and very robust antenna prototypes.

It is important to note that the final five-patch array antenna has a bandwidth $> 20\%$, which makes it perfectly valid for Ultra Wide Band (UWB) applications. It also has a minimum gain of 5.13 dBi and a maximum of 9.37 dBi, which are excellent for such a compact antenna, which is only 82.5 mm long.

6.1 Future Lines of Work

The presented work could be continued by researching in the following fields (some of them have already been explored but not finished):

Architecture Design for Localization

The presented multi-technology architecture was defined as a general methodology for implementing RTLSs. Its specific use for solving different localization problems, with more experimental results in different scenarios, or adapting to even more complex multi-technology localization problems, could be future lines of work.

It would also be interesting to improve the architecture to be highly scalar and with a high availability, to make it compatible with those critical applications or where a huge number of targets to be located is required.

At the time of writing this thesis, the proposed architecture was already implemented and extended many times, using a RESTful APIs (instead of TCP sockets) and other client authentication techniques (such as OAuth 2.0). It was used for implementing different commercial RTLS software variants, making use of measurements coming from WiFi or

Bluetooth Low Energy (BLE) portable devices (i.e. smartphones or tags), maybe in combination with visual camera sensor information.

Customized Hardware Design for Localization

In the case of RFID technology used for localization, we could continue testing the constructed hardware prototype in real scenarios, and with different kind of antennas and tags.

The construction of the RF switch box hardware could also be reduced by using a smaller PCB layout and a plastic enclosure, to make it more affordable and ready for mass production in other non industrial sectors, where an antenna concentrator would be required to cover a greater localization area.

Design of Antennas for Localization

In this matter, there are still many possibilities and combination techniques that could be applied for the creation of new ECP based antenna models to improve the performance of WSNs in different senses:

- By adapting better to different indoor scenarios structures and heights, where different polarizations or coverage zones are required, or other scenarios where we want a more omnidirectional antenna (i.e. small portable tags) where the antennas would be implemented with a smaller ground plane. We would thus obtain, in any direction, an electrical field which is closer to the mean field of the antenna.
- We could also group the radiating patches in two or more rows, making a rectangular grid array antenna, which would have a better directivity, for solving those cases where the antenna pointing should be more precise, as for instance, for communications between repeater equipment.
- We could also try to build ECP based antenna models with multi-frequency support, for working within the 2.45 and 5 GHz bands, for instance.

Algorithm Design for Localization

We have also started a research collaboration with the UC3M university related to location algorithms based on Bayesian filtering but, unfortunately, this work is unfinished and remains as a future line of work. Its main objective was the design of new state-space models that take advantage of Magnetic, Angular Rate, and Gravity (MARG) sensors with the objective of improving the tracking performance of a well-known state-space model. And with the help of a new Maximum A Posteriori (MAP) model selection methodology we would compare the proposed state-space models.

In this particular case we used a UWB WSN platform with Time-Difference-of-Arrival (TDoA) observations starting with a well-known coordinated turn dynamic model, and we improved this model in two steps, by adding vibration information (easy obtained from a general purpose accelerometer or vibration sensor) and magnetic azimut information (that can trivially be extracted from a general purpose magnetometer). Both were very low-cost Microelectromechanical Systems (MEMS) sensors that could be added to any moving target device to easily improve its tracking accuracy. We proposed a MAP model selection methodology in order to compare the three proposed state-space models, but were unable to conclude the experimental evaluation of this comparison using real measurements, already obtained in a laboratory.

Appendix A

Resumen de la Tesis

A.1 Introducción

El problema de la localización en el interior de edificios ha ido adquiriendo cada vez más importancia en los últimos años debido a la enorme demanda de nuevos servicios basados en localización (LBSs), que han ido apareciendo en la industria en sectores de todo tipo como eHealth [108], puntos de interés [78, 141], smart-parking y smart-cities, seguridad y emergencias, logística o control industrial [67, 126], entre otros. Estos sistemas habitualmente se basan en la implementación de redes de sensores inalámbricos (WSN) capaces de transmitir o recibir señales de radio (RF) para localizar dispositivos móviles, generalmente adheridos a vehículos, personas o animales.

Tiempo atrás, en 2006, cuando comenzamos nuestra investigación en el campo de la localización en interiores, no había demasiadas alternativas disponibles para implementar WSNs, válidos para resolver los problemas de localización en interiores a un coste reducido (lo cual es siempre deseable para cualquier sistema RTLS). En aquella época los telefonos celuláres no eran tan *smart* y los portatiles eran mucho menos comunicativos que hoy en día, y sólo un par de tecnologías inalámbricas como Bluetooth y a veces WiFi estaban implementados en estos dispositivos y eran, por tanto, válidos para interactuar con las WSNs. En aquel momento también empezaron a aparecer otras tecnologías inalámbricas como ZigBee o ultra-sonidos, principalmente disponibles en forma de kits de desarrollo o de evaluación, que podrían ser utilizados para implementar sistemas RTLS, por ejemplo, de una forma no centralizada.

Hoy en día, existe una continua evolución de estas tecnologías y continúan apareciendo infinidad de dispositivos portátiles que también podrían utilizarse para fines de localización. Un aumento en el número de teléfonos móviles, smart-phones, Personal Digital Assistants (PDAs), tablets y portátiles, que están equipados con todavía más interfaces de radio, como pueden ser WiFi, BLE, Enhanced GPS (E-GPS) y Assisted GPS (A-GPS), Near Field Communication (NFC) e incluso sensores inerciales, como acelerómetros, giroscópio o compás

digital. Por suerte, podemos sacar provecho de estas características para mejorar la localización en interiores.

También han ido apareciendo multiples plataformas de prototipado rápido que permiten combinar múltiples transceptores de RF y sensores, como son Arduino [43], Raspberry Pi [53], Teensy [55], panSTamp [52], BeagleBone [44] y muchos otros.

Mientras que en exteriores los sistemas satelitales basados en tecnologías como GPS funcionan correctamente en la mayoría de entornos, la localización en interiores todavía plantea múltiples retos y no es una tarea sencilla de resolver. Principalmente aparecen problemas de propagación debido a los reflejos y rebotes de las señales en las estructuras de los edificios, pero también debido a atenuaciones y apantallamientos ocasionados generalmente por gente en movimiento. Para resolver estos problemas es necesario implementar las redes de sensores utilizando una o varias tecnologías inalámbricas (como pueden ser WiFi, ZigBee o Bluetooth), y como ya se ha comentado, algunas de ellas están disponibles en terminales inalámbricos como smartphones o tablets. Por otra parte, también es necesario el uso de múltiples algoritmos y técnicas de localización, para filtrar y posiblemente combinar los datos de estas tecnologías, permitiendo obtener así sistemas de localización en tiempo real (RTLS) robustos y con la mayor precisión posible.

Desde el punto de vista del hardware todas las alternativas disponibles presentan algo en común. Restringen de forma importante el diseño del sistema RTLS, debido principalmente a las particularidades de cada una de ellas a la hora de obtener las mediciones, debido a que utilizan diferentes interfaces de acceso. Por otra parte, también aparecen restricciones debidas a *drivers* de software y librerías, que habitualmente están disponibles para una única plataforma y sistema operativo específicos, que residen en el equipo encargado de recolectar y procesar las mediciones obtenidas. Los sistemas operativos abiertos normalmente permiten un acceso más amplio y libre a todos los niveles, siendo por tanto lo más apropiados para implementar estos sistemas, pero en algunas situaciones el hardware sólo está soportado en alguna plataforma cerrada.

Por estas razones, las primeras plataformas RTLS comenzaron implementándose usando un único lenguaje de programación y sistema operativo (el que ofreciera la máxima funcionalidad) y típicamente estábamos forzados a portar (reescribir o adaptar) los algoritmos de localización (en ocasiones muy complejos) a alguna plataforma específica. Estos algoritmos normalmente ya habían sido implementados en algún entorno de simulación de alto nivel como Matlab y este proceso habitualmente tenía que repetirse para nuevos RTLSs usando diferentes WSNs y tecnologías, lo que supone un gran coste y consumo de tiempo.

Además, cuando necesitamos conseguir datos desde una WSN en tiempo real, estamos forzados a adaptar los algoritmos de localización ya implementados, para que trabajen con datos de tipo *streaming*, y esto requiere de un enrome esfuerzo por parte de programadores e investigadores. Esta gente normalmente está acostumbrada a trabajar con datos de tipo off-line,

ficheros de log o datos simulados y cargados en memoria de golpe, en un único paso.

La aproximación más usual en la actualidad para resolver todos estos problemas, es la implementación de sistemas de localización híbridos que soporten múltiples tecnologías simultáneamente. No obstante, el desarrollo de estos sistemas lleva implícito una gran complejidad. Una de las alternativas comúnmente aceptada es la implementación de una arquitectura de software para localización, que ofrece múltiples ventajas. En primer lugar, permite minimizar el número de restricciones multi-plataforma y multi-tecnología a la hora de acceder a distintos tipos de dispositivos hardware. En segundo lugar, se facilitan tareas comunes como la recolección y almacenamiento de las mediciones de los sensores. Además, se proveen mecanismos para insertar y recuperar datos de localización así como gestión de usuarios o manejo de múltiples sistemas de mapas y coordenadas.

A.2 Principales Objetivos

El proposito de este trabajo está centrado en el diseño e implementación de arquitecturasde software válidas para resolver sistemas de localización en tiempo real (RTLS) complejos y multi-tecnología. Empezamos con una limitada arquitectura mono-tecnología que fue seguida por una multi-tecnología mucho más avanzada. No solo nos centramos en una arquitectura en cuanto al hardware sino también en el diseño de software y en experimentos RTLS. Hemos realizado mútiples campañas de mediciones tanto en escenarios interiores como exteriores en los que hemos evaluado experimentalmente varios sistemas WSN usando tecnologías Bluetooth, ZigBee y GPS. También hemos aportado otros resultados basados en simulación, usando las tecnologías WiFi, ZigBee, RDID, UWB y sensores inerciales.

Gracias a la colaboracion con el departamenteo de Teoría de la Señal y las Comunicaciones de la Universidad Carlos III de Madrid (UC3M) pudimos evaluar nuestros desarrollos de arquitecturas de software y campañas de mediciones en múltiples escenarios y con avanzadas técnicas de localización [63–66].

También colaboramos con el Institute of Communications and Radio-Frequency Engineering, Vienna University of Technology, Austria, donde trabajamos en la construcción y despliegue de un testbed Long Term Evolution (LTE) [81–84, 115, 134] y ganamos experiencia en el diseño de bajo nivel de circuitos de radiofrecuencia (RF). Como resultado de esta colaboración, implementamos una solución de hardware personalizada, para reducir los costes de hardware de sistemas RTLS basados en la tecnología RFID UHF. Con la solución propuesta conseguimos minimizar el número de lectores RFID (hardware costoso) necesarios para una infraestructura de localización, maximizando el número de antenas que se pueden conectar a cada lector (hasta 16 antenas con nuestra propuesta).

Y finalmente, también se realizó una colaboración con Julio Brégains, del Grupo de

Tecnología Electrónica y Comunicaciones (GTEC) de la Universidad de A Coruña, el cual posee una gran trayectoria en el campo de investigación en el diseño de antenas. Simulamos e implementamos nuevos modelos numéricos de antenas mediante arrays lineales de ECPs. Proporcionamos metodologías de diseño que permiten adaptar estos modelos a diferentes rangos de frecuencia haciendo que, por tanto, no sean sólo válidas para mejorar el rendimiento de comunicaciones WLAN, sino también de sistemas de localización basados en WSNs usando transceptores de radio (todos aquellos que no utilizan sistemas ópticos o sensores acústicos). También aplicamos una nueva técnica de prototipado durante la construcción de dos prototipos de antenas, que nos permiten su construcción de una forma más precisa, obteniéndose así una aproximación lo más parecida al modelo numérico planteado mediante simulación, y evitándose en definitiva la mayoría de imprecisiones debidas a la manipulación humana durante su construcción.

A.3 Conclusiones

Las principales contribuciones de este trabajo se muestran en los Capítulos 2, 3, 4 y 5:

1. La primera contribución se muestra en el Capítulo 2, donde presentamos una arquitectura distribuida y multi-capa para implementar sistemas RTLS. Sólo soporta una única tecnología inalámbrica (a la vez), basada en una red de sensores inalámbricos (WSN), anchors situados en posiciones fijas localizando a nodos móviles. Además, sólo es compatible con mediciones RSS. La arquitectura propuesta está dividida en cuatro capas: Una capa *Physical* encargada de acceder al hardware de la WSN a bajo nivel: una capa *Location* donde se implementa un algoritmo de localziación (que puede remotamente acceder a la capa *Physical*), y finalmente, otras dos capas (*Proxy* y *Wrapper*) encargadas de abstraer completamente las comunicaciones entre las anteriores dos capas.

   Los principales beneficios conseguidos son la reusabilidad e independencia de las capas implementadas, que pueden estar implementadas en diferentes lenguajes de programación, y el posible acceso remoto al hardware de diferentes WSN desde múltiples clientes *Location*.

   Además, mostramos dos posibles formas de implementar la capa física, mediante tecnologías inalámbricas Bluetooth y ZigBee, así como diferentes implementaciónes y experimentos para ambas tecnologías:

   - Para el caso de la tecnología Bluetooth, aplicamos la arquitectura para la evaluación experimental de un método basado en filtrado Bayesiano. En particular para la estimación conjunta de la posición de un *target* y del exponente path-loss de un modelo de propagación. Mediante la ejecución de diferentes experimentos

mostramos, en primer lugar, el efecto de elegir un exponente diferente para un anchor arbitrario mientras el resto lo estiman correctamente. Y en segundo lugar, como el filtro de partículas implementado es capaz de adaptarse automáticamente a cambios de propagación, entre una situación LOS y NLOS, mientras la posición del target está siendo estimada simultáneamente.

- Para el caso de la tecnología ZigBee, aplicamos la arquitectura para la evaluación experimental de métodos de filtrado Bayesiano, en este caso para la localización de targets en entornos mixtos interiores/exteriores usando las tecnologías ZigBee y GPS.

  Los resultados experimentales mostrados presentan la localización de un dispositivo móvil siguiendo tres trayectorias diferentes. Se muestran las diferencias entre usar únicamente GPS para las estimaciones (mediante un filtro de Kalman), o bien usar mediciones RSS de ZigBee de forma combinada con posiciones de GPS (mediante un algoritmo SIS). Finalmente, también se aportan algunos resultados mediante simulación, mostrando un experimento de localización híbrido interior/exterior. En este caso se usan cada de las dos tecnologías cuando están disponibles (en cada parte del trayecto). Por tanto, en ocasiones se utilizan de forma individual y, a veces, de forma combinada.

- Y para el caso de ZigBee también proponemos un método para incrementar el número de mediciones RSS disponibles en cada momento, mediante la obtención de datos RSS entre cada pareja de nodos de la red ZigBee.

  Con mediciones RSS extra obtenidas para todos los enlaces entre anchors, podríamos mejorar el rendimiento de ciertos algoritmos de localización, posiblemente implementando alguna técnica de recalibración dinámica o ajuste de parámetro del modelo de propagación.

  Mostramos como este método se podría implementar en sólo tres pasos usando hardware de prototipado común como puede ser Arduino y módulos XBee. Toda la lógica propuesta y el formato de tramas de salida son completamente compatibles con el formato propuesto para la capa *Physical* de esta primera arquitectura de software.

2. La segunda contribución se muestra en el Capítulo 3, donde se presenta una arquitectura RTLS que soporta múltiples tecnologías inalámbricas simultáneamente, múltiples sistemas de coordenadas y de mapas, fusión de datos a bajo y a alto nivel, protección y seguridad tanto de la información como en el acceso de usuarios a la pataforma, entre otras. La arquitectura también fue diseñada para proporcionar mecanismos que de una forma sencilla permitan registrar nuevo hardware, añadir algoritmos de localización e

implementar LBSs de tipo cliente que consuman dichos datos de localización.

La arquitectura también proporciona un Application Programming Interface (API) completo con múltiples métodos para la comunicación con clientes externos. Los usuarios que se encarguen de proporcionar hardware de localización, podrán dar de alta en la plataforma diferentes redes de nodos anchor, móviles y sensores de todo tipo, que podrán ser utilizados de forma transparente por los desarrolladores de algoritmos de localización. Estos algoritmos de localización podrán usar datos en crudo de una o más tecnologías y nodos, de los disponibles en el sistema, para proporcionar estimaciones de localización. Y estas estimaciones, a su vez, estarán disponibles en tiempo-real (o *off-line*, para el acceso a históricos de datos) por aplicaciones cliente de localización.

Las principales ventajas que ofrece la arquitectura propuesta, en comparación con otras soluciones monolíticas, es que permite una generalización y reutilización de cada parte del sistema para resolver nuevos LBSs y aplicaciones, reduciendo por tanto los costes de desarrollo. Además, su soporte para dar de alta nuevas tecnologías de sensores de forma totalmente personalizada (utilizando casi cualquier combinación de tipos de datos) le permiten resolver, en principio, cualquier tipo de sistema RTLS multi-tecnología. Esta solución difiere de otras de propósito específico, que pueden ser más caras y difíciles implementar en la práctica, sobre todo para resolver casos diferentes a los que fueron concebidas.

Para demostrar la aplicabilidad de la arquitectura, mostramos dos casos de estudio basados en experimentos reales. En primer lugar, un dispositivo taget multi-tecnología basado en ZigBee, UWB y un acelerómetro. Y en segundo lugar, un escenario de fingerprinting implementado con WiFi y utilizando un smartphone Android convencional. Finalmente, mediante varios resultados de validación se demuestra el alto rendimiento de la plataforma a la hora de localización un dispositivo multi-tecnología ZigBee y UWB en un escenario interior. Estos resultados fueron obtenidos mediante mediciones generadas en un entorno de simulación de alto-nivel (Matlab), haciendo uso del API real e interfaz de red de la plataforma, mediante el uso de llamadas de red.

3. La tercera contribución se presenta en el Capítulo 4, donde se muestra una solución conjunta software y hardware para mejorar la diversidad de antenas de lectores RFID, que puede ser aprovechada para implementar sistemas RTLS más baratos, basados en tecnología RFID. La solución propuesta está basada en una infraestructura de lectores RFID UHF (siguiendo el estándar ISO 18000-6C EPC Gen2), y presenta un buen compromiso entre complejidad de hardware y software, mientras se minimizan los costes. Los switches RF propuestos permiten multiplicar por un factor de hasta cuatro el número de antenas disponibles en cada lector RFID, se trata de una solución auto-alimentada

mediante el propio lector RFID, y controlada directamente mediante su puerto de uso general (GPIO). Los resultados muestran una comparativa de costes que demuestra que nuestra solución es económica.

4. Finalmente, la cuarta contribución de esta tesis se presenta en el Capítulo 5, centrado en el diseño de nuevos modelos numéricos de antenas, basados en topologías de tipo array lineal de ECPs. Presentamos tres modelos numéricos que cubren diferentes bandas de frecuencia, compatibles con los estandares de WiFi y WiMAX. También proponemos una metodología de diseño para cada una de las antenas, que permiten adaptarlas de forma trivial a todo tipo de rangos de frecuencia. Por tanto, en principio, podrían ser válidas para la mayoría de WLANs y WSNs en escenarios tanto interiores como exteriores.

   En primer lugar, empezamos con el diseño de una antena ECP compuesta por un único parche radiante, que representa la base de las otras dos antenas propuestas. La segunda antena es de tipo array, está compuesta por tres parches distribuidos de forma trigonal y principalmente consigue una mayor cobertura horizontal respecto al anterior caso. Finalmente, diseñamos un modelo numérico de antena formado por cinco parches que están distribuidos de forma coplanar sobre un mismo substrato.

   Para la evaluación experimental de los modelos numéricos propuestos, decidimos construir varios prototipos de antenas y medir su rendimiento en una cámara anecóica.

   Durante la construcción de estos dos prototipos de antenas, se usaron PCBs Rogers (de la serie RO4000) que fueron dispuestos en paralelo y adheridos a una carcasa de plástico (usada como soporte), con la ayuda de tornillos de nylon y arandelas de acero (con un grosor muy preciso). De esta forma fuimos capaces de mantener de forma muy estable y precisa la orientación y separación de los tres PCBs que forman cada antena. Todos estos factores, el uso de materiales RF (de alta calidad), de las partes utilizadas (con grosores muy precisos) y de la técnica de montaje propuesta, resultaron en prototipos de antenas de alta calidad y muy robustos.

   Es importante resaltar que la antena obtenida de cinco partes tiene un ancho de banda tan bueno, $> 20\%$, que es perfectamente válida para aplicaciones de UWB. También se consiguen unas ganancias, mínima de 5.13 dBi y máxima de 9.37 dBi que son excelentes para una antena tan contenida, cuya dimensión máxima es de 82.5 mm.

A.4 Líneas Futuras de Trabajo

El trabajo presentado podría continuarse mediante la investigación en los siguientes campos (algunos de ellos ya han sido explorados pero no finalizados):

Diseño de Arquitecturas para Localización

La arquitectura multi-tecnología presentada fue definida como una metodología general para implementar sistemas RTLS. Su uso específico para resolver diferentes problemas de localización, con más resultados experimentales en diferentes escenarios, o adaptándose a sistemas de localización incluso más complejos que los inicialmente contemplados, podrían representar lineas futuras de trabajo.

También sería muy interesante mejorar la arquitectura para que pudiera ser altamente escalable y de alta disponibilidad, de forma que soportase aplicaciones críticas o aquellas donde se requiriese localizar un elevado número de *targets*.

En el momento de escribir esta tesis, la arquitectura propuesta ya ha sido implementada y extendida múltiples veces, usando por ejemplo, un API RESTful (en lugar de sockets TCP) y añadiendo otras técnicas de autenticación de clientes (como OAuth 2.0). Fue utilizada para implementar diferentes variantes de software comercial RTLS, haciendo uso de mediciones provenientes de dispositivos móviles con WiFi o BLE (ej. smartphones o tags), usándose posiblemente en combinación con información sensorial procedente de cámaras de vídeo.

Diseño de Hardware Personalizado para Localización

Para el caso de uso de la tecnología RFID para localización, podríamos utilizar los prototipos de hardware construidos en diferentes escenarios reales, con diferentes tipos de antenas y tags (dispuestos sobre diferentes *targets*).

La construcción del switch RF podría reducirse de tamaño utilizando PCBs más pequeños y abaratarse todavía más utilizando carcasas de plástico. De esta forma se harían más asequibles y apropiados para su posible producción en masa orientada a entornos no industriales, donde igualmente se requieren concentradores de antenas para conseguir una mayor cobertura de localización.

Diseño de Antenas para Localización

Existen múltiples posibilidades y técnicas de combinación que podrían ser aplicadas a la creación de nuevos modelos de antenas basados en ECPs, para mejorar el rendimiento de redes WSNs en varios sentidos:

- Adaptándolos mejor a diferentes estructuras y alturas de escenarios interiores, donde diferentes polarizaciones y zonas de cobertura son requeridas, o para otros escenarios donde se requieren antenas más omnidireccionales (ej. pequeños tags portátiles). En este caso las antenas podrían implementarse con planos de tierra de menor tamaño, para obtener un campo eléctrico en cualquier dirección lo más cercano al campo medio de toda la antena.

- También podríamos agrupar los parches radiantes en filas, construyendo agrupaciones de mallado rectangular, para mejorar la directividad, en casos en que se requieran usos en donde el apuntamiento de las antenas deba ser más preciso, como por ejemplo radiación entre puntos repetidores.

- También se podría intentar construir modelos multi-frecuencia, que trabajen simultáneamente en las bandas de 2.45 y 5 GHz, por ejemplo.

Diseño de Algoritmos para Localización

También hemos iniciado una colaboración con la Universidad Carlos III de Madrid (UC3M) relacionada con algoritmos de localización basados en filtrado Bayesiano pero, desafortunadamente, el trabajo realizado no fue completado y permanece como una linea futura de trabajo. Su principal objetivo era el diseño de nuevos modelo de espacio-estado que saquen provecho de sensores MARG, con el objetivo de mejorar la precisión de localización obtenida con respecto a un conocido modelo de espacio-estado de referencia. Y con la ayuda de una nueva metodología MAP de selección de modelos, compararíamos los modelos espacio-estado propuestos.

En nuestro caso en particular utilizamos una plataforma de UWB y observaciones de TDoA, partiendo de un conocido modelo dinámico *coordinated turn*, y mejoramos este modelo en dos pasos, añadiendo en primer lugar información de vibración (que se puede obtener de cualquier sensor de propósito general como un acelerómetro o sensor de vibración) e información de azimut magnético (que también se puede extraer de forma trivial de cualquier magnetómetro de propósito general). Ambos sensores MEMS son de muy bajo coste y se podrían añadir a casi cualquier target móvil para fácilmente mejorar la precisión de localización. También propusimos e implementamos una metodología MAP de selección de modelos para poder comparar los tres modelos espacio-estado propuestos, pero no pudimos concluir su evaluación experimental utilizando mediciones reales, ya obtenidas en un laboratorio.

Appendix B

List of Acronyms

AoA Angle-of-Arrival

AFH Adaptive Frequency Hopping

API Application Programming Interface

AUT Antenna Under Test

BLE Bluetooth Low Energy

BOM Bill of Materials

CDF Cumulative Distribution Function

DSP Digital Signal Processor

ECP Electromagnetically Coupled Patch

A-GPS Assisted GPS

E-GPS Enhanced GPS

EDR Enhanced Data Rate

FDTD Finite Difference Time Domain

FPGA Field Programmable Gate Array

GNSS Global Navigation Satellite System

GoF Gang-of-Four

GPIO General-Purpose Input/Output

GPS Global Positioning System

HCI Host Controller Interface

HTTP Hypertext Transfer Protocol

IC Integrated Circuit

IoT Internet of Things

IPS Indoor Positioning System

ISM Industrial, Scientific and Medical

ISO International Standard Organization

JCR Journal Citation Report

JSON JavaScript Object Notation

JSON-RPC JavaScript Object Notation Remote Procedure Call

JVM Java Virtual Machine

LBS Location Based Service

LLRP Low-Level Reader Protocol

LOS Line of Sight

LTE Long Term Evolution

MAC Media Access Control

MAP Maximum A Posteriori

MARG Magnetic, Angular Rate, and Gravity

MEMS Microelectromechanical Systems

MMSE Minimum Mean Square Estimate

NFC Near Field Communication

NMEA National Marine Electronics Association

NLOS Non Line of Sight

PAN Personal Area Network

PCB Printed Circuit Board

PDA Personal Digital Assistant

PF Particle Filter

PNA Personal Navigation Assistant

PoE Power-over-Ethernet

RF Radio Frequency

RHCP Right Hand Circularly Polarized

RFID Radio-Frequency Identification

RPC Remote Procedure Call

RTLS Real-Time Location System

RSS Received Signal Strength

RSSI Received Signal Strength Indicator

SIG Special Interest Group

SIR Sequential Importance Resampling

SIS Sequential Importance Sampling

SMA SubMiniature version A

SMC Sequential Monte Carlo

SOAP Simple Object Access Protocol

SOC System-on-a-Chip

SSL Secure Sockets Layer

TDoA Time-Difference-of-Arrival

TLS Transport Layer Security

ToA Time-of-Arrival

UHF Ultra-High Frequency

UWB Ultra Wide Band

WLAN Wireless Local Area Network

WPAN Wireless Personal Area Network

WSN Wireless Sensor Network

XML eXtensible Markup Language

XML-RPC eXtensible Markup Language Remote Procedure Call

Yaml YAML Ain't Another Markup Language

Bibliography

[1] (2015, aug.) Atmel ATmega168 AVR MCU
http://www.atmel.com/Images/doc8161.pdf

[2] (2015, aug.) Agilent PNA E8361A network analizer
http://www.keysight.com/en/pd-1000004621%3Aepsg%3Apro-pn-E8361A/pna-series

[3] (2015, aug.) AIRcable long-range Bluetooth devices
http://www.aircable.net/products

[4] (2015, aug.) Bluetooth SIG
http://www.bluetooth.org

[5] (2015, aug.) Cambridge Silicon Radio (CSR)
http://www.csr.com

[6] (2015, aug.) ECal N4691B electronic calibration kit
http://literature.cdn.keysight.com/litweb/pdf/5963-3743E.pdf?id=1000030554:epsg:dow

[7] (2015, aug.) ETS-Lindgren 3117 antenna
http://www.ets-lindgren.com/pdf/3117.pdf

[8] (2015, aug.) Ekahau Real Time Location System
http://www.ekahau.com/real-time-location-system/technology

[9] (2015, aug.) Facebook Graph API
https://developers.facebook.com/docs/graph-api

[10] (2015, aug.) FourSquare API
https://developer.foursquare.com/

[11] (2015, aug.) Hittite HMC544 / HMC544E RF switches
http://www.hittite.com/products/view.html/view/HMC544

[12] (2015, aug.) ISO/IEC 19762-5:2008 - Automatic Identification and Data Capture (AIDC) techniques. Harmonized vocabulary. Part 5: Locating systems
http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50718

[13] (2015, aug.) ISO/IEC 24730-1:2006 - Real-Time Locating Systems (RTLS). Part 1: Application Program Interface (API)
http://www.iso.org/iso/catalogue_detail.htm?csnumber=38840

[14] (2015, aug.) Impinj True3D technology

http://www.impinj.com/products/tag-chips/monza-4qt/

[15] (2015, aug.) Impinj Speedway Revolution RFID Readers

http://www.impinj.com/products/readers/

[16] (2015, aug.) JSON-Benchmark: Which parser is fastest on Android?

http://code.google.com/p/json-benchmark/

[17] (2015, aug.) JSON-Smart vs All Other Java JSON Parser

https://code.google.com/p/json-smart/wiki/Benchmark

[18] (2015, aug.) Updated iPhone JSON benchmarks

http://blog.soff.es/updated-iphone-json-benchmarks

[19] (2015, aug.) JSON versus PLIST, the ultimate showdown

http://www.cocoanetics.com/2011/03/json-versus-plist-the-ultimate-showdown

[20] (2015, aug.) Laird Technologies S8658-PR antenna

https://support.impinj.com/hc/en-us/article_attachments/200774718/IPJ-A1000-A1001-EU1_Antenna_Spec.pdf

[21] (2015, aug.) MicaZ

http://www.memsic.com/products/wireless-sensor-networks/wireless-modules.html

[22] (2015, aug.) Mojix eLocation

http://www.mojix.com/products/elocation.php

[23] (2015, aug.) OpenID

http://openid.net

[24] (2015, aug.) Plus Location Systems

http://pluslocation.com/

[25] (2015, aug.) ProgrammableWeb: Is JSON the Developer's Choice?

http://www.programmableweb.com/news/json-developers-choice/2010/08/11

[26] (2015, aug.) ProgrammableWeb: 1 in 5 APIs Say "Bye XML"

http://www.programmableweb.com/news/1-5-apis-say-bye-xml/2011/05/25

[27] (2015, aug.) ProgrammableWeb: 81 News APIs: Digg, FanFeedr and ClearForest

http://www.programmableweb.com/news/81-news-apis-digg-fanfeedr-and-clearforest/2012/02/01

[28] (2015, aug.) Rogers RO4000 laminates

http://www.rogerscorp.com/acs/producttypes/9/RO4000-Laminates.aspx

[29] (2015, aug.) Secure Sockets Layer (SSL) Protocol Version 3.0

http://tools.ietf.org/html/rfc6101

[30] (2015, aug.) SEMCAD X. Reference Guide

http://www.eecs.wsu.edu/~schneidj/Tmp/semcad-x-manual.pdf

[31] (2015, aug.) Transport Layer Security (TLS) Protocol Version 1.2

http://tools.ietf.org/html/rfc5246

[32] (2015, aug.) UPM RFID DogBone passive tag

http://www.rfidtags.com/documents/product/UPM-RFID-DogBone_M4_datasheet.pdf

[33] (2015, aug.) UPM RFID Frog3D passive tag

http://www.rfidtags.com/documents/product/UPM-RFID-Frog%203D_datasheet.pdf

[34] (2015, aug.) Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: IEEE Std 802.11-2012

http:

//eclass.uoa.gr/modules/document/file.php/D428/%CE%91%CF%81%CE%B8%CF%81%CE%B1/WiFi/Standard_802.11-2012.pdf

[35] (2015, aug.) Developing the IEEE 802.16 WirelessMAN Standard for Wireless Metropolitan Area Networks

http://www.ieee802.org/16/

[36] (2015, aug.) XBee wireless modules

http://www.digi.com/xbee

[37] (2015, aug.) XBee Series 1 802.15.4 wireless modules

http://www.digi.com/products/xbee-rf-solutions/modules/xbee-series1-module

[38] (2015, aug.) XBee Series 1 802.15.4 datasheet

http://www.digi.com/support/getasset?fn=90000982&tp=3

[39] (2015, aug.) Extensible Markup Language (XML)

http://www.w3.org/XML

[40] (2015, aug.) YAML Ain't Markup Language

http://www.yaml.org

[41] (2015, aug.) ZigBee Alliance

http://www.zigbee.org

[42] (2015, aug.) Apache XML-RPC Java implementation

http://ws.apache.org/xmlrpc/

[43] (2015, aug.) Arduino

http://www.arduino.cc

[44] (2015, aug.) BeagleBone

http://beagleboard.org/bone

[45] (2015, aug.) BlueZ Official Linux Bluetooth protocol stack

http://www.bluez.org

[46] (2015, aug.) FTDI FT232R USB UART IC

http://www.ftdichip.com/Products/ICs/FT232R.htm

[47] (2015, aug.) Google Gson

http://code.google.com/p/google-gson

[48] (2015, aug.) Jackson Java JSON-processor

http://wiki.fasterxml.com/JacksonHome

[49] (2015, aug.) JSON-lib

http://json-lib.sourceforge.net

[50] (2015, aug.) JSON-RPC
http://www.jsonrpc.org/

[51] (2015, aug.) Orbital/FR AL-4806-3C positioner controller
www.orbitfr.com/files/AL-48060_SeriesDataSheet.pdf

[52] (2015, aug.) panSTamp
http://www.panstamp.com

[53] (2015, aug.) Raspberry Pi
http://www.raspberrypi.org

[54] (2015, aug.) Simple Object Access Protocol (SOAP)
http://www.w3schools.com/webservices/ws_soap_intro.asp

[55] (2015, aug.) Teensy USB Development Board
https://www.pjrc.com/teensy

[56] (2015, aug.) Introducing Tumblr's New API
http://engineering.tumblr.com/post/7541361718/introducing-tumblrs-new-api

[57] (2015, aug.) UNIX Timestamp format
http://www.unixtimestamp.com

[58] (2015, aug.) XML-RPC implementations
http://directory.xmlrpc.com/implementations/

[59] (2015, aug.) XML-RPC lightweight C and C++ implementations
http://xmlrpc-c.sourceforge.net

[60] (2015, aug.) XML-RPC PHP implementations
http://phpxmlrpc.sourceforge.net

[61] (2015, aug.) XML-RPC
http://www.xmlrpc.com

[62] (2015, aug.) XMLRPC-EPI C implementation
http://xmlrpc-epi.sourceforge.net

[63] K. Achutegui, L. Martino, J. Rodas, C. Escudero, and J. Míguez, "A Multi-Model Particle Filtering Algorithm for Indoor Tracking of Mobile Terminals Using RSS Data," in *IEEE Control Applications (CCA) & Intelligent Control (ISIC)*, jul. 2009, pp. 1702–1707, doi: 10.1109/CCA.2009.5280960.

[64] K. Achutegui, J. Rodas, C. Escudero, and J. Míguez, "A Model-Switching Sequential Monte Carlo Algorithm for Indoor Tracking with Experimental RSS," in *Indoor Positioning and Indoor Navigation (IPIN)*, sep. 2010, pp. 1–8, doi: 10.1109/IPIN.2010.5648053.

[65] K. Achutegui, J. Rodas, C. J. Escudero, and J. Míguez, "Bayesian Filtering Methods for Target Tracking in Mixed Indoor/Outdoor Environments," in *ICST Mobile Lightweight Wireless Systems (MOBILIGHT)*, vol. 81, may. 2011, pp. 169–185, doi: 10.1007/978-3-642-29479-2_13.

[66] K. Achutegui, J. Míguez, J. Rodas, and C. Escudero, "A Multi-Model Sequential Monte Carlo

Methodology for Indoor Tracking: Algorithms and Experimental Results," *ELSEVIER Signal Processing*, vol. 92, no. 11, pp. 2594–2613, 2012, doi: 10.1016/J.SIGPRO.2012.03.017.

[67] M. Arikawa, S. Konomi, and K. Ohnishi, "Navitime: Supporting Pedestrian Navigation in the Real World," *IEEE Pervasive Computing*, vol. 6, no. 3, pp. 21–29, 2007, doi: 10.1109/MPRV.2007.61.

[68] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A Tutorial on Particle Filters for On-line Non-linear/Non-Gaussian Bayesian Tracking," *IEEE Transactions of Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002, doi: 10.1109/78.978374.

[69] A. Bahillo, J. Prieto, S. Mazuelas, R. Lorenzo, P. Fernaández, and E. Abril, "E-Field Assessment Errors Caused by the Human Body on Localization Systems," in *IEEE Vehicular Technology Conference (VTC 2010-Spring)*, may. 2010, pp. 1–5, doi: 10.1109/VETECS.2010.5493636.

[70] P. Bahl and V. N. Padmanabhan, "RADAR: An In-Building RF-based User Location and Tracking System," in *Computer and Communications Societies (INFOCOM)*, mar. 2000, pp. 775–784, doi: 10.1109/INFCOM.2000.832252.

[71] ——, "Enhancements to the RADAR user location and tracking system," Microsoft Research, Tech. Rep. MSR-TR-2000-12, feb. 2000
http://research.microsoft.com/apps/pubs/default.aspx?id=69861

[72] C. Balanis, *Antenna Theory. Analysis and Design. 3rd Edition*. Wiley, 2005. ISBN: 978-0471667827.

[73] ——, *Modern Antenna Handbook*. Wiley, 2008. ISBN: 978-0470036341.

[74] R. Battiti, T. L. Nhat, and A. Villani, "Location-Aware Computing: A Neural Network Model For Determining Location In Wireless LANs," Department of Information and Communication Technology, University of Trento, Tech. Rep. DIT-02-0083, feb. 2002
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.20.2366

[75] A. Boukerche, H. A. B. F. Oliveira, E. F. Nakamura, and A. A. F. Loureiro, "Localization Systems for Wireless Sensor Networks," *IEEE Wireless Communications*, vol. 14, no. 6, pp. 6–12, 2007, doi: 10.1109/MWC.2007.4407221.

[76] J. Brégains, G. Franceschetti, A. Roederer, and F. Ares, "New Toroidal Beam Antennas for WLAN Communications," *IEEE Transactions on Antennas and Propagation*, vol. 55, no. 2, pp. 389–398, 2007, doi: 10.1109/TAP.2006.889796.

[77] J. C. Brégains, "Análisis, Síntesis y Control de Diagramas de Radiación Generados por Distribuciones Continuas y Discretas Utilizando Algoritmos Estocásticos y Redes Neuronales Artificiales. Aplicaciones," Ph.D. dissertation, University of Santiago de Compostela, Department of Applied Physics, Electromagnetism Area, 2007.

[78] E. Bruns, B. Brombach, T. Zeidler, and O. Bimber, "Enabling Mobile Phones to Support Large-Scale Museum Guidance," *IEEE Multimedia*, vol. 14, no. 2, pp. 16–25, 2007, doi: 10.1109/MMUL.2007.33.

[79] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, *Pattern-Oriented Software Architecture Volume 1: A System of Patterns*. Wiley, 1996. ISBN: 978-0471958697.

[80] ——, *Pattern-Oriented Software Architecture: A System of Patterns*. Wiley, 2008. ISBN: 978-8126516117.

[81] S. Caban, A. Disslbacher-Fink, J. A. García-Naya, and M. Rupp, "Synchronization of Wireless Radio Testbed Measurements," *IEEE Instrumentation and Measurement Technology Conference (I2MTC)*, pp. 1–4, may 2011, doi: 10.1109/IMTC.2011.5944089.

[82] S. Caban, J. Garcia-Naya, and M. Rupp, "Measuring the Physical Layer Performance of Wireless Communication Systems: Part 33 in a Series of Tutorials on Instrumentation and Measurement," *IEEE Instrumentation & Measurement Magazine*, vol. 14, no. 5, pp. 8–17, 2011, doi: 10.1109/MIM.2011.6041377.

[83] S. Caban, J. Rodas, and J. Garcia-Naya, "A Methodology for Repeatable, Off-Line, Closed-Loop Wireless Communication System Measurements at Very High Velocities of Up to 560 km/h," in *Instrumentation and Measurement Technology Conference (I2MTC)*, may. 2011, pp. 1–5, doi: 10.1109/IMTC.2011.5944019.

[84] S. Caban, R. Nissel, M. Lerch, and M. Rupp, "Controlled OFDM Measurements at Extreme Velocities," in *Extreme Conference on Communication and Computing (ExtremeCom)*, aug. 2014.

[85] J. Carpenter, P. Clifford, and P. Fearnhead, "Improved Particle Filter for Nonlinear Problems," *IEE Proceedings Radar, Sonar and Navigation*, vol. 146, no. 1, pp. 2–7, 1999, doi: 10.1049/IP-RSN:19990255.

[86] M. D. Coster, S. Mattheussen, M. Klepal, M. Weyn, and G. Ergeerts, "GeCSen - A Generic and Cross-Platform Sensor Framework for LocON," in *Mobile Ubiquitous Computing, Systems, Services and Technologies*, 2010, pp. 21–26.

[87] S. Couronne, N. Hadaschik, M. Fassbinder, T. von der Grun, M. Weyn, M. Klepal, Widyawan, and T. Denis, "LocON - A Platform for an Inter-Working of Embedded Localisation and Communication Systems," in *Sensor, Mesh and Ad Hoc Communications and Networks Workshops (SECON)*, jun. 2009, pp. 1–3, doi: 10.1109/SAHCNW.2009.5172944.

[88] D. Crisan, *Particle Filters - A Theoretical Perspective*, ser. Statistics for Engineering and Information Science, A. Doucet, N. de Freitas, and N. Gordon, Eds. Springer New York, 2001. ISBN: 978-1441928870.

[89] F. Dieter, J. Hightower, L. Liao, D. Schulz, and G. Borriello, "Bayesian Filtering for Location Estimation," *IEEE Pervasive Computing*, vol. 2, no. 3, pp. 24–33, 2003, doi: 10.1109/MPRV.2003.1228524.

[90] P. M. Djurić, J. H. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. F. Bugallo, and J. Míguez, "Particle Filtering," *IEEE Signal Processing Magazine*, vol. 20, no. 5, pp. 19–38, 2003, doi: 10.1109/MSP.2003.1236770.

[91] A. Doucet, S. Godsill, and C. Adrieu, "On Sequential Monte Carlo Sampling Methods for Bayesian Filtering," *Statistics and Computing*, vol. 10, no. 3, pp. 197–208, 2000, doi: 10.1023/A:1008935410038.

[92] A. Doucet, N. Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*. Springer,

2001. ISBN: 978-1475734379.

[93] M. R. Ehsani, M. D. Sullivan, T. L. Zimmerman, and T. Stombaugh, "Evaluating the Dynamic Accuracy of Low-Cost GPS Receivers," in *American Society of Agricultural and Biological Engineers (ASAE)*, jul. 2003, doi: 10.13031/2013.15019.

[94] R. Elliot, *Antenna Theory & Design, Revised Edition*. Wiley, 2003. ISBN: 978-0471449966.

[95] E. Elnahrawy, L. Xiaoyan, and R. P. Martin, "The Limits of Localization using Signal Strength: a Comparative Study," in *IEEE Sensor and Ad Hoc Communications and Networks*, oct. 2004, pp. 406–414, doi: 10.1109/SAHCN.2004.1381942.

[96] O. Fresnedo, D. Iglesia, and C. J. Escudero, "Bluetooth Inquiry Procedure: Optimization and Influence of the Number of Devices," in *IASTED Communications Systems and Networks*, 2007, pp. 205–209.

[97] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994. ISBN: 978-0201633610.

[98] D. Garlan and M. Shaw, "An Introduction to Software Architecture," *Advances in Software Engineering and Knowledge Engineering*, vol. 1, pp. 1–40, 1993.

[99] M. Goran and R. Richton, "Geolocation and Assisted GPS," *IEEE Communications Magazine*, vol. 34, no. 2, pp. 123–125, 2001, doi: 10.1109/2.901174.

[100] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel Approach to Nonlinear/Non-Gaussian Bayesian State Estimation," *IEE Proceedings-F*, vol. 140, no. 2, pp. 107–113, 1993.

[101] Y. Guo, A. Paez, R. Sadeghzadeh, and S. Barton, "A Circular Patch Antenna for Radio LAN's," *Antennas and Propagation, IEEE Transactions on*, vol. 45, pp. 177–178, 1997, doi: 10.1109/8.554256.

[102] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P. Nordlund, "Particle Filters for Positioning, Navigation and Tracking," *IEEE Transactions Signal Processing*, vol. 50, no. 2, pp. 425–437, feb. 2002, doi: 10.1109/78.978396.

[103] M. Haller, M. Billinghurst, and B. Thomas, *Emerging Technologies of Augmented Reality: Interfaces and Design*. IGI Global, 2006. ISBN: 978-1599040660.

[104] R. Hansen and B. Thomsen, "Efficient and Accurate WLAN Positioning with Weighted Graphs," *Mobile Lightweight Wireless Systems*, vol. 13, pp. 372–386, 2009, doi: 10.1007/978-3-642-03819-8_35.

[105] J. Hightower and G. Borriello, "Location Systems for Ubiquitous Computing," *IEEE Computer Magazine*, vol. 34, no. 8, pp. 57–66, 2001, doi: 10.1109/2.940014.

[106] J. Hightower, G. Borriello, and R. Want, "SpotON: An Indoor 3D Location Sensing Technology Based on RF Signal Strength," University of Washington, Tech. Rep. UW CSE 00-02-02, 2000
http://bib-di.inf.puc-rio.br/~endler/courses/Mobile/papers/LPS/spoton.pdf

[107] J. Hightower, B. Brumitt, and G. Borriello, "The Location Stack: a Layered Model for Location in Ubiquitous Computing," in *Mobile Computing Systems and Applications. 4th IEEE Workshop*

on, 2002, pp. 22–28, doi: 10.1109/MCSA.2002.1017482.

[108] B. Jit, D. Zhang, G. Qiao, V. Foo, Q. Qiu, and P. Yap, "A System for Activity Monitoring and Patient Tracking in a Smart Hospital," *Smart Homes and Beyond (ICOST)*, pp. 196–204, 2006, doi: 10.1.1.474.4488.

[109] K. Kaemarungsi, "Distribution of WLAN Received Signal Strength Indication for Indoor Location Determination," in *Wireless Pervasive Computing*, jan. 2006, doi: 10.1109/ISWPC.2006.1613601.

[110] K. Kaemarungsi and P. Krishnamurthy, "Modeling of Indoor Positioning Systems Based on Location Fingerprinting," in *Computer and Communications Societies (INFOCOM)*, vol. 2, may. 2004, pp. 1012–1022, doi: 10.1109/INFCOM.2004.1356988.

[111] E. D. Kaplan, *Understanding GPS: Principles and Applications*. Artech House Publishers, 2005. ISBN: 978-1580538947.

[112] T. King, T. Haenselmann, and W. Effelsberg, "Deployment, Calibration, and Measurement Factors for Position Errors in 802.11-Based Indoor Positioning Systems," *Location and Context-Awareness*, vol. 4718, pp. 17–34, 2007, doi: 10.1007/978-3-540-75160-1_2.

[113] M. Klepal, R. Mathur, A. McGibney, and D. Pesch, "Influence of People Shadowing on Optimal Deployment of WLAN Access Points," in *IEEE Vehicular Technology Conference (VTC 2004-Fall)*, vol. 6, sep. 2004, pp. 4516–4520, doi: 10.1109/VETECF.2004.1404934.

[114] C. Laoudias, D. Eliades, P. Kemppi, C. Panayiotou, and M. Polycarpou, "Indoor Localization Using Neural Networks with Location Fingerprints," *Artificial Neural Networks (ICANN)*, vol. 5769, pp. 954–963, 2009, doi: 10.1007/978-3-642-04277-5_96.

[115] M. Lerch, S. Caban, M. Mayer, and M. Rupp, "The Vienna MIMO Testbed: Evaluation of Future Mobile Communication Techniques," *Intel Technology Journal, 4G Wireless Communications: Real World Aspects and Tools*, vol. 18, no. 3, pp. 58–69, 2014.

[116] B. Li, J. Salter, A. G. Dempster, and C. Rizos, "Indoor Positioning Techniques Based on Wireless LAN," in *Wireless Broadmand and Ultra Wideband Communications*, 2006, pp. 13–16.

[117] X. Li, "RSS-Based Location Estimation with Unknown Pathloss Model," *IEEE Transactions on Wireless Communications*, vol. 5, no. 12, pp. 3626–3633, 2006, doi: 10.1109/TWC.2006.256985.

[118] X. N. Low, W. Toh, and Z. N. Chen, "Broadband Suspended Plate Antenna for WiFi/WiMAX Applications," in *Information, Communications Signal Processing*, dec. 2007, pp. 1–5, doi: 10.1109/ICICS.2007.4449555.

[119] E. Lutz, D. Cygan, M. Dippold, F. Dolainsky, and W. Papake, "The Land Mobile Satellite Communication Channel - Recording, Statistics, and Channel Model," *IEEE Transactions on Vehicular Technology*, vol. 40, no. 2, pp. 375–386, 1991, doi: 10.1109/25.289418.

[120] J. Míguez, "Analysis of Parallelizable Resampling Algorithms for Particle Filtering," *Signal Processing*, vol. 87, no. 12, pp. 3155–3174, 2007, doi: 10.1016/J.SIGPRO.2007.06.011.

[121] H. Millner, P. Gulden, M. Weyn, M. Fassbinder, and A. Casaca, "LocON Deliverable D4.2: Description of the LocON protocol," Symeo, CIT, CEA-LETI, INOV, Artesis, Fraunhofer IIS,

Tech. Rep., mar. 2009

http://www.locon-eu.com/Deliverables/DeliverableD42_prot.pdf

[122] C. Montes, J. Rodas, L. Castedo, and J. Brégains, "Diseño y Prototipado de una Agrupación de Antenas UltraWideBand para Aplicaciones WiFi y WiMAX," in *XXVIII Simposium Nacional de la Unión Científica Internacional de Radio (URSI 2013)*, Santiago (Spain), sep. 2013.

[123] C. Morelli, M. Nicoli, V. Rampa, and U. Spagnolini, "Hidden Markov Models for Radio Localization in Mixed LOS/NLOS Conditions," *IEEE Transactions on Signal Processing*, vol. 55, no. 4, pp. 1525–1542, 2007, doi: 10.1109/TSP.2006.889978.

[124] A. Narzullaev, Y. Park, and H. Jung, "Accurate Signal Strength Prediction Based Positioning for Indoor WLAN Systems," in *IEEE Position, Location and Navigation Symposium*, may. 2008, pp. 685–688, doi: 10.1109/PLANS.2008.4569989.

[125] L. M. Ni, Y. Liu, Y. C. Lau, and A. P. Patil, "LANDMARC: Indoor Location Sensing Using Active RFID," in *IEEE Pervasive Computing and Communications (PerCom)*, mar. 2003, pp. 407–415, doi: 10.1109/PERCOM.2003.1192765.

[126] M. O'Connor, T. Bell, G. Elkaim, and B. Parkinson, "Automatic Steering of Farm Vehicles Using GPS," in *Precision Agriculture*, jun. 1996, p. 767–778.

[127] T. Rappaport, *Wireless Communications: Principles and Practice*. Prentice Hall, 2002. ISBN: 007-6092011736.

[128] J. Rodas and C. J. Escudero, "Multi-layer Architecture for Location Systems Based on Wireless Sensor Networks," in *Signals, Systems and Computers (ASILOMAR)*, nov. 2009, pp. 270–274, doi: 10.1109/ACSSC.2009.5470107.

[129] ——, "Cross Measurement Process with a ZigBee Sensor Network," in *Signals, Systems and Computers (ASILOMAR)*, nov. 2009, pp. 279–283, doi: 10.1109/ACSSC.2009.5470104.

[130] ——, "Joint Estimation of Position and Channel Propagation Model Parameters in a Bluetooth Network," in *IEEE International Conference on Communications Workshops (ICC)*, jun. 2009, pp. 1–5, doi: 10.1109/ICCW.2009.5207993.

[131] J. Rodas, V. Barral, and C. Escudero, "Architecture for Multi-Technology Real-Time Location Systems," *Sensors*, vol. 13, no. 2, pp. 2220–2253, 2010, doi: 10.3390/s130202220.

[132] J. Rodas, J. Brégains, L. Castedo, and F. Ares, "Diseño de una Antena Conformada Eficiente para Aplicaciones WiMAX," in *XXV Simposium Nacional de la Unión Científica Internacional de Radio (URSI 2010)*, Bilbao (Spain), sep. 2010.

[133] J. Rodas, V. Barral, C. Escudero, and R. Langwieser, "A Solution for Optimizing Costs and Improving Diversity of RFID Readers," in *Systems, Signals and Image Processing (IWSSIP)*, apr. 2012, pp. 84–88.

[134] J. Rodriguez-Piñeiro, M. Lerch, J. Garcia-Naya, S. Caban, M. Rupp, and L. Castedo, "Emulating Extreme Velocities of Mobile LTE Receivers in the Downlink," *EURASIP Journal on Wireless Communications and Networking*, vol. 2015, no. 106, 2015, doi: 10.1186/s13638-015-0343-0.

[135] T. Roos, P. Myllymäki, H. Tirri, P. Misikangas, and J. Sievänen, "A Probabilistic Approach to WLAN User Location Estimation," *International Journal of Wireless Information Networks*, vol. 9, no. 3, pp. 155–164, 2002, doi: 10.1023/A:1016003126882.

[136] J. Roy and M. Thomas, "Investigations on a New Proximity Coupled Dual-Frequency Microstrip Antenna for Wireless Communication," *Microwave Review*, vol. 13, no. 1, pp. 12–15, 2007.

[137] S. Saha, K. Chauhuri, D. Sanghi, and P. Bhagwat, "Location Determination of a Mobile Device Using IEEE 802.11b Access Point Signals," in *IEEE Wireless Communications and Networking (WCNC)*, vol. 3, may. 2003, pp. 1987–1992, doi: 10.1109/WCNC.2003.1200692.

[138] T. K. Sarkar, Z. Ji, K. Kim, A. Medouri, and M. Salazar-Palma, "A Survey of Various Propagation Models for Mobile Communication," *IEEE Antennas and Propagation Magazine*, vol. 45, no. 3, pp. 51–82, 2003, doi: 10.1109/MAP.2003.1232163.

[139] J. Schiller and A. Voisard, *Location-based Services*. Elsevier, 2004. ISBN: 978-1493303786.

[140] M. Shaw and D. Garlan, *Software Architecture: Perspectives on an Emerging Discipline*. Prentice Hall Englewood Cliffs, NJ, 1996. ISBN: 978-0131829572.

[141] R. Simon, P. Frohlich, and H. Anegg, "Beyond Location Based - The Spatially Aware Mobile Phone," *Web and Wireless Geographical Information Systems*, vol. 4295, pp. 12–21, 2006, doi: 10.1007/11935148_2.

[142] R. W. Stevens and S. A. Rago, *Advanced Programming in the UNIX(R) Environment (2Nd Edition)*. Addison-Wesley Professional, 2005. ISBN: 978-0321525949.

[143] S. Sullivan, "Comparing JSON Libraries," Portland Java User Group, Tech. Rep., jul. 2011
http://www.slideshare.net/sullis/comparing-json-libraries-july-19-2011

[144] P. Tarrio, A. Bernardos, and J. Casar, "An RSS Localization Method Based on Parametric Channel Models," in *Sensor Technologies and Applications (SensorComm)*, oct. 2007, pp. 265–270, doi: 10.1109/SENSORCOMM.2007.4394932.

[145] M. Weyn, "Opportunistic Seamless Localization," Ph.D. dissertation, University of Antwerp, mar. 2011.

[146] H. P. William, P. F. Brian, A. T. Saul, and T. V. William, *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992. ISBN: 978-0521431088.

[147] M. G. Wing, A. Eklund, and L. D. Kellogg, "Consumer-Grade Global Positioning System (GPS) Accuracy and Reliability," *Journal of Forestry*, vol. 103, no. 4, pp. 169–173, 2005.

[148] K.-L. Wong, *Compact and Broadband Microstrip Antennas*. Willey, 2002. ISBN: 978-0471417170.

[149] C. Wu, L. Fu, and F. Lian, "WLAN Location Determination in e-Home via Support Vector Classification," *IEEE Networking, Sensing and Control*, vol. 2, pp. 1026–1031, 2004, doi: 10.1109/ICNSC.2004.1297088.

[150] R. Yamasaki, A. Ogino, T. Tamaki, T. Uta, N. Matsuzawa, and T. Kato, "TDOA Location System for IEEE 802.11b WLAN," *IEEE Wireless Communications and Networking Conference*, vol. 4,

pp. 2338–2343, 2007, doi: 10.1109/WCNC.2005.1424880.